

国外电子信息类系列教材

Statistical and Adaptive Signal processing

统计与自适应信号处理

（英文改编版）

Dimitris G.Manolakis
［美］ Vinay K.Ingle 著
Stephen M.Kogon

阔永红 改编

Electronic Information



西安电子科技大学出版社
http://www.xduph.com



■. 内容简介

本书介绍了统计与自适应信号处理的基本概念和应用，包括随机序列分析、谱估计以及自适应滤波等内容。本书可作为电子、通信、自动化、电机、生物医学和机械工程等专业研究生作为教材或教学参考书，也可作为广大工程技术人员的自学读本或参考用书。

■. 作者简介

Dimitris G. Manolakis：于希腊雅典大学获得物理学士学位和电气工程博士学位，现任美国麻省林肯实验室研究员；曾在Riverside研究所任主任研究员，并曾在雅典大学、美国东北大学、波士顿学院、沃切斯特理工学院任教。

Vinay K. Ingle：于伦斯勒理工学院获得电气和计算机工程的博士学位，曾在多所大学讲授过信号处理课程，具有丰富的研究经历；1981年加入美国东北大学，目前在电气工程和计算机系任职。

Stephen M. Kogon：于佐治亚理工学院获得电气工程博士学位，现任美国麻省林肯实验室研究员；曾就职于Raytheon公司、波士顿大学和佐治亚技术研究所。





国外电子信息类系列教材

Statistical and Adaptive Signal Processing

统计与自适应信号处理(英文改编版)

Spectral Estimation, Signal Modeling and Adaptive Filtering

Dimitris G.Manolakis

Massachusetts Institute of Technology

Lincoln Laboratory

Vinay K.Ingle

Northeastern University

Stephen M.Kogon

Massachusetts Institute of Technology

Lincoln Laboratory

阎永红　改编

西安电子科技大学出版社

图书在版编目(CIP)数据

统计与自适应信号处理=Statistical and Adaptive Signal processing：英文名/(美)马诺拉可斯(Manolakis, D.G.),
(美)英格勒(Ingle, V.K.), (美)哥根(Kogon, S.M.)著. —西安：西安电子科技大学出版社，2012.8
ISBN 978-7-5606-2848-6

Ⅰ. ① 统… Ⅱ. ① 马… ② 英… ③ 哥… Ⅲ. ① 统计信号—信号处理—英文
② 自适应控制—信号处理—英文 Ⅳ. ① TN911.72

中国版本图书馆 CIP 数据核字(2012)第 149062 号

Dimitris G .Manolakis, Vinay K.Ingle, Stephen M.Kogen
Statistical and Adaptive Signal Processing, Spectral Estimation, Signal Modeling ,Adaptive Filtering and Array Processing.
ISBN：0-07-040051-2

策　　划　曹媛媛
责任编辑　曹媛媛　马晓娟
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　西安文化彩印厂
版　　次　2012 年 8 月第 1 版　　2012 年 8 月第 1 次印刷
开　　本　880 毫米×1230 毫米　1/16　印　张　24.75
字　　数　645 千字
印　　数　1～3000 册
定　　价　56.00 元
ISBN 978-7-5606-2848-6/TN・0664
XDUP　3140001 -1

ABOUT THE AUTHORS

DIMITRIS G.MANOLAKIS, a native of Greece, received his education (B.S. in physics and Ph.D. in electrical engineering) from the University of Athens, Greece. He is currently a member of the technical staff and MIT Lincoln Laboratory, in Lexington, Massachusetts. Previously, he was a Principal Member, Research Staff, at Riverside Research Institute. Dr. Manolakis has taught at the University of Athens, Northeastern University, Boston College, and Worcester Polytechnic Institure; and he is coauthor of the textbook *Digital Signal Processing: Principles, Algorithms, and Applications* (Prentice-Hall, 1996, 3d ed.). His research experience and interests include the areas of digital signal processing, adaptive filtering, array processing, pattern recognition, and radar systems.

VINAY K.INGLE is Associate Professor of Electrical and Computer Engineering at Northeastern University. He received his Ph.D. in electrical and computer engineering from Rensselaer Polytechnic Institute in 1981. He has broad research experience and has taught courses on topics including signal and image processing, stochastic processes, and estimation theory. Professor Ingle is coauthor of the textbooks *DSP Laboratory Using the ADSP-2101 Microprocessor* (Prentice-Hall, 1991) and *DSP Using Matlab* (PWS Publishing Co., Boston, 1996).

STEPHEN M.KOGON received the Ph.D. degree in electrical engineering from Georgia Institute of Technology. He is currently a member of the technical staff at MIT Lincoln Laboratory in Lexington, Massachusetts. Previously, he has been associated with Raytheon Co., Boston College, and Georgia Tech Research Institute. His research interests are in the areas of adaptive processing, array signal processing, radar, and statistical signal modeling.

One must learn by doing the thing;
for though you think you know it
You have no certainty, until you try.
—Sophocles, *Trachiniae*

PREFACE

The principal goal of this book is to provide a unified introduction to the theory, implementation, and applications of statistical and adaptive signal processing methods. We have focused on the key topics of spectral estimation, signal modeling, adaptive filtering, whose selection was based on the grounds of theoretical value and practical importance. The book has been primarily written with students and instructors in mind. The principal objectives are to provide an introduction to basic concepts and methodologies that can provide the foundation for further study, research, and application to new problems. To achieve these goals, we have focused on topics that we consider fundamental and have either multiple or important applications.

Approach and prerequisites

The adopted approach is intended to help both students and practicing engineers understand the fundamental mathematical principles underlying the operation of a method, appreciate its inherent limitations, and provide sufficient details for its practical implementation. The academic flavor of this book has been influenced by our teaching whereas its practical character has been shaped by our research and development activities in both academia and industry. The mathematical treatment throughout this book has been kept at a level that is within the grasp of upper-level undergraduate students, graduate students, and practicing electrical engineers with a background in digital signal processing, probability theory, and linear algebra.

Organization of the book

Chapter 1 introduces the basic concepts and applications of statistical and adaptive signal processing and provides an overview of the book. Chapters 2 introduce some basic concepts of estimation theory. Chapter 3 provides a treatment of parametric linear signal models (both deterministic and stochastic) in the time and frequency domains. Chapter 4 presents the most practical methods for the estimation of correlation and spectral densities. Chapter 5 provides a detailed study of the theoretical properties of optimum filters, assuming that the relevant signals can be modeled as stochastic processes with known statistical properties; and Chapter 6 contains algorithms and structures for optimum filtering, signal modeling, and prediction. Chapter 7 introduces the principle of least-squares estimation and its application to the design of practical filters and predictors. Chapters 8 and 9 use the theoretical work in Chapters 3, 5 and 6 and the practical methods in Chapter 7 to develop, evaluate, and apply practical techniques for signal modeling, adaptive filtering,

Theory and practice

It is our belief that sound theoretical understanding goes hand-in-hand with practical implementation and application to real-world problems. Therefore, the book includes a large number of computer experiments that illustrate important concepts and help the reader to easily implement the various methods. Every chapter includes examples, problems, and computer experiments that facilitate the comprehension of the material. To help the reader understand the theoretical basis and limitations of the various methods and apply them to real-world problems, we provide MATLAB functions for all major algorithms and examples illustrating their use.

Feedback

Although we are fully aware that there always exists room for improvement, we believe that this book is a big step forward for an introductory textbook in statistical and adaptive signal processing. However, as engineers, we know that every search for the optimum requires the will to change and quest for additional improvement. Thus, we would appreciate feedback from teachers, students, and engineers using this book for self-study at `vingle@lynx.neu.edu`.

Acknowledgments

We are indebted to a number of individuals who have contributed in different, but important, ways to the shaping of our knowledge in general and the preparation of this book in particular. In this respect we wish to extend our appreciation to C. Caroubalos, G. Carayannis, J. Makhoul (DGM), M. Schetzen (VKI), and J. Holder, S. Krich, and D. Williams (SMK). We are grateful to E. Baranoski, G. Borsari, J. Schodorf, S. Smith, A. Steinhardt, and J. Ward for helping us to improve the presentation in various parts of the book.

We express our sincere gratitude to Kevin Donohue, University of Kentucky; Amro El-Jaroudi, University of Pittsburgh; Edward A. Lee, University of California-Berkeley; Ray Liu, University of Maryland; Randy Moses, The Ohio State University; and Kristina Ropella, Marquette University, for their constructive and helpful reviews.

Lynn Cox persuaded us to choose McGraw-Hill, Inc., as our publisher, and we have not regretted that decision. We are grateful to Lynn for her enthusiasm and her influence in shaping the scope and the objectives of our book. The fine team at McGraw-Hill, including Michelle Flomenhoft, Catherine Fields, Betsy Jones, and Nina Kreiden, has made the publication of this book an exciting and pleasant experience. We also thank N. Bulock and Mathworks, Inc., for promptly providing various versions of MATLAB and A. Turcotte for helping with some of the drawings in the book.

Last, but not least, we would like to express our sincere appreciation to our families for their full-fledged support and understanding over the past several years. We fully realize that the completion of such a project would not be possible without their continual sustenance and encouragement.

Dimitris G. Manolakis
Vinay K. Ingle
Stephen M. Kogon

Contents

CHAPTER 1 Introduction 1
1.1 Random Signals 1
1.2 Spectral Estimation 3
1.3 Signal Modeling 6
1.4 Adaptive Filtering 9
1.4.1 Applications of Adaptive Filters 9
1.4.2 Features of Adaptive Filters 14
1.5 Organization of the Book 16
CHAPTER 2 Random Sequences 17
2.1 Discrete-Time Stochastic Processes 17
2.1.1 Description Using Probability Functions 19
2.1.2 Second-Order Statistical Description 19
2.1.3 Stationarity 21
2.1.4 Ergodicity 24
2.1.5 Random Signal Variability 26
2.1.6 Frequency-Domain Description of Stationary Processes 27
2.2 Linear Systems with Stationary Random Inputs 33
2.2.1 Time-Domain Analysis 33
2.2.2 Frequency-Domain Analysis 35
2.2.3 Random Signal Memory 36
2.2.4 General Correlation Matrices 37
2.2.5 Correlation Matrices from Random Processes 40
2.3 Innovations Representation of Random Vectors 41
2.4 Principles of Estimation Theory 44
2.4.1 Properties of Estimators 44
2.4.2 Estimation of Mean 46
2.4.3 Estimation of Variance 49
2.5 Summary 51
Problems 52
CHAPTER 3 Linear Signal Models 57
3.1 Introduction 57
3.1.1 Linear Nonparametric Signal Models 58
3.1.2 Parametric Pole-Zero Signal Models 60
3.1.3 Mixed Processes and Wold Decomposition 62
3.2 All-Pole Models 63
3.2.1 Model Properties 63
3.2.2 All-Pole Modeling and Linear Prediction 69
3.2.3 Autoregressive Models 69
3.2.4 Lower-Order Models 70
3.3 All-Zero Models 77
3.3.1 Model Properties 77
3.3.2 Moving-Average Models 78
3.3.3 Lower-Order Models 78

3.4 Pole-Zero Models ... 80
3.4.1 Model Properties ... 81
3.4.2 Autoregressive Moving-Average Models ... 83
3.4.3 The First-Order Pole-Zero Model: PZ(1, 1) ... 83
3.4.4 Summary and Dualities ... 84
3.5 Summary ... 85
Problems ... 85
CHAPTER 4 Nonparametric Power Spectrum Estimation ... 89
4.1 Spectral Analysis of Deterministic Signals ... 89
4.1.1 Effect of Signal Sampling ... 91
4.1.2 Windowing, Periodic Extension, and Extrapolation ... 91
4.1.3 Effect of Spectrum Sampling ... 92
4.1.4 Effects of Windowing: Leakage and Loss of Resolution ... 95
4.1.5 Summary ... 100
4.2 Estimation of the Autocorrelation of Stationary Random Signals ... 101
4.3 Estimation of the Power Spectrum of Stationary Random Signals ... 103
4.3.1 Power Spectrum Estimation Using the Periodogram ... 103
4.3.2 Power Spectrum Estimation by Smoothing a Single Periodogram—The Blackman-Tukey Method 112
4.3.3 Power Spectrum Estimation by Averaging Multiple Periodograms—The Welch-Bartlett Method .. 117
4.3.4 Some Practical Considerations and Examples ... 121
4.4 Multitaper Power Spectrum Estimation ... 125
4.5 Summary ... 130
Problems ... 130
CHAPTER 5 Optimum Linear Filters ... 136
5.1 Optimum Signal Estimation ... 136
5.2 Linear Mean Square Error Estimation ... 138
5.2.1 Error Performance Surface ... 139
5.2.2 Derivation of the Linear MMSE Estimator ... 142
5.2.3 Principal-Component Analysis of the Optimum Linear Estimator ... 143
5.2.4 Geometric Interpretations and the Principle of Orthogonality ... 145
5.2.5 Summary and Further Properties ... 146
5.3 Optimum Finite Impulse Response Filters ... 147
5.3.1 Design and Properties ... 148
5.3.2 Optimum FIR Filters for Stationary Processes ... 149
5.3.3 Frequency-Domain Interpretations ... 153
5.4 Linear Prediction ... 154
5.4.1 Linear Signal Estimation ... 154
5.4.2 Forward Linear Prediction ... 156
5.4.3 Backward Linear Prediction ... 156
5.4.4 Stationary Processes ... 157
5.4.5 Properties ... 160
5.5 Optimum Infinite Impulse Response Filters ... 162
5.5.1 Noncausal IIR Filters ... 163
5.5.2 Causal IIR Filters ... 163
5.5.3 Filtering of Additive Noise ... 166
5.5.4 Linear Prediction Using the Infinite Past—Whitening ... 171
5.6 Inverse Filtering and Deconvolution ... 173
5.7 Summary ... 176

Problems 177
CHAPTER 6 Algorithms and Structures for Optimum Linear Filters 182
6.1 Fundamentals of Order-Recursive Algorithms 182
6.1.1 Matrix Partitioning and Optimum Nesting 183
6.1.2 Inversion of Partitioned Hermitian Matrices 184
6.1.3 Levinson Recursion for the Optimum Estimator 186
6.1.4 Order-Recursive Computation of the LDLH Decomposition 188
6.1.5 Order-Recursive Computation of the Optimum Estimate 189
6.2 Interpretations of Algorithmic Quantities 191
6.2.1 Innovations and Backward Prediction 192
6.2.2 Partial Correlation 192
6.2.3 Order Decomposition of the Optimum Estimate 193
6.2.4 Gram-Schmidt Orthogonalization 194
6.3 Order-Recursive Algorithms for Optimum FIR Filters 195
6.3.1 Order-Recursive Computation of the Optimum Filter 196
6.3.2 Lattice-Ladder Structure 199
6.3.3 Simplifications for Stationary Stochastic Processes 201
6.4 Algorithms of Levinson and Levinson-Durbin 202
6.5 Lattice Structures for Optimum Fir Filters And Predictors 208
6.5.1 Lattice-Ladder Structures 219
6.5.2 Some Properties and Interpretations 211
6.5.3 Parameter Conversions 212
6.6 Summary 214
Problems 215
CHAPTER 7 Least-Squares Filtering and Prediction 220
7.1 The Principle of Least Squares 220
7.2 Linear Least-Squares Error Estimation 221
7.2.1 Derivation of the Normal Equations 222
7.2.2 Statistical Properties of Least-Squares Estimaters 227
7.3 Least-Squares FIR Filters 231
7.4 Linear Least-Squares Signal Estimation 235
7.4.1 Signal Estimation and Linear Prediction 235
7.4.2 Combined Forward and Backward Linear Prediction(FBLP) 237
7.4.3 Narrowband Interference Cancelation 238
7.5 LS Computations Using the Normal Equations 241
7.5.1 Linear LSE Estimation 241
7.5.2 LSE FIR Filtering and Prediction 244
7.6 Summary 245
Problems 246
CHAPTER 8 Signal Modeling and Parametric Spectral Estimation 250
8.1 The Modeling Process: Theory and Practice 250
8.2 Estimation of All-Pole Models 253
8.2.1 Direct Structures 253
8.2.2 Lattice Structures 261
8.2.3 Maximum Entropy Method 263
8.2.4 Excitations with Line Spectra 264
8.3 Estimation Of Pole-Zero Models 265
8.3.1 Known Excitation 265

8.3.2 Unknown Excitation 266
8.4 Applications 267
8.4.1 Spectral Estimation 267
8.4.2 Speech Modeling 269
8.5 Harmonic Models and Frequency Estimation Techniques 271
8.5.1 Harmonic Model 272
8.5.2 Pisarenko Harmonic Decomposition 275
8.5.3 MUSIC Algorithm 276
8.5.4 Minimum-Norm Method 278
8.5.5 ESPRIT Algorithm 281
8.6 Summary 285
Problems 286
CHAPTER 9 Adaptive Filters 291
9.1 Typical Applications of Adaptive Filters 291
9.1.1 Echo Cancelation in Communications 291
9.1.2 Linear Predictive Coding 293
9.1.3 Noise Cancelation 295
9.2 Principles of Adaptive Filters 296
9.2.1 Features of Adaptive Filters 296
9.2.2 Optimum versus Adaptive Filters 297
9.2.3 Stability and Steady-State Performance of Adaptive Filters 301
9.2.4 Some Practical Considerations 304
9.3 Method of Steepest Descent 305
9.4 Least-Mean-Square Adaptive Filters 312
9.4.1 Derivation 312
9.4.2 Adaptation in a Stationary SOE 314
9.4.3 Summary and Design Guidelines 320
9.4.4 Applications of the LMS Algorithm 323
9.4.5 Some Practical Considerations 331
9.5 Recursive Least-Squares Adaptive Filters 333
9.5.1 LS Adaptive Filters 333
9.5.2 Conventional Recursive Least-Squares Algorithm 337
9.5.3 Some Practical Considerations 338
9.5.4 Convergence and Performance Analysis 340
9.6 Fast RLS Algorithms for FIR Filtering 344
9.6.1 Fast Fixed-Order RLS FIR Filters 346
9.6.2 RLS Lattice-Ladder Filters 351
9.6.3 RLS Lattice-Ladder Filters Using Error Feedback Updatings 354
9.7 Tracking Performance of Adaptive Algorithms 356
9.7.1 Approaches for Nonstationary SOE 356
9.7.2 Preliminaries in Performance Analysis 359
9.7.3 LMS Algorithm 362
9.7.4 RLS Algorithm with Exponential Forgetting 367
9.7.5 Comparison of Tracking Performance 372
9.8 Summary 372
Problems 373

CHAPTER 1

Introduction

This book is an introduction to the theory and algorithms used for the analysis and processing of random signals and their applications to real-world problems. The fundamental characteristic of random signals is captured in the following statement: Although random signals are evolving in time in an unpredictable manner, their average statistical properties exhibit considerable regularity. This provides the ground for the description of random signals using statistical averages instead of explicit equations. When we deal with random signals, the main objectives are the statistical description, modeling, and exploitation of the dependence between the values of one or more discrete-time signals and their application to theoretical and practical problems.

Random signals are described mathematically by using the theory of probability, random variables, and stochastic processes. However, in practice we deal with random signals by using statistical techniques. Within this framework we can develop, at least in principle, theoretically *optimum signal processing* methods that can inspire the development and can serve to evaluate the performance of practical *statistical signal processing* techniques. The area of *adaptive signal processing* involves the use of optimum and statistical signal processing techniques to design signal processing systems that can modify their characteristics, during normal operation (usually in real time), to achieve a clearly predefined application-dependent objective.

The purpose of this chapter is twofold: to illustrate the nature of random signals with some typical examples and to introduce the three major application areas treated in this book: *spectral estimation*, *signal modeling and adaptive filtering*. Throughout the book, the emphasis is on the application of techniques to actual problems in which the theoretical framework provides a foundation to motivate the selection of a specific method.

1.1 Random Signals

A *discrete-time signal* or *time series* is a set of observations taken sequentially in time, space, or some other independent variable. Examples occur in various areas, including engineering, natural sciences, economics, social sciences, and medicine.

A discrete-time signal $x(n)$ is basically a sequence of real or complex numbers called samples. Although the integer index n may represent any physical variable (e.g., time, distance), we shall generally refer to it as *time*. Furthermore, in this book we consider only time series with observations occurring at equally spaced intervals of time.

Discrete-time signals can arise in several ways. Very often, a discrete-time signal is obtained by periodically sampling a continuous-time signal, that is, $x(n) = x_c(nT)$, where $T = 1/F_s$ (seconds) is the sampling period and F_s (samples per second or hertz) is the sampling frequency. At other times, the samples of a discrete-time signal are obtained by accumulating some quantity (which does not have an instantaneous value) over equal intervals of time, for example, the number of cars per day traveling on a certain road. Finally, some signals are inherently discrete-time, for example, daily stock market prices. *Throughout the book, except if otherwise stated, the terms* signal, time series, *or* sequence *will be used to refer to a discrete-time signal.*

The key characteristics of a time series are that the observations are *ordered* in time and that adjacent observations are *dependent* (related). To see graphically the relation between the samples of a signal that are l sampling intervals away, we plot the points $\{x(n), x(n+l)\}$ for $0 \le n \le N-1-l$, where N is the length of the data record. The resulting graph is known as the *l lag scatter plot*. This is illustrated in Figure 1.1, which shows a speech signal and two scatter plots that demonstrate the correlation between successive samples. We note that for adjacent samples the data points fall close to a straight line with a positive slope. This implies high correlation because every sample is followed by a sample with about the same amplitude. In contrast, samples that are 20 sampling intervals apart are much less correlated because the points in the scatter plot are randomly spread.

When successive observations of the series are dependent, we may use past observations to predict future values. If the prediction is exact, the series is said to be *deterministic*. However, in most practical situations we cannot predict a time series exactly. Such time series are called *random* or *stochastic*, and the degree of their predictability is

determined by the dependence between consecutive observations. The ultimate case of randomness occurs when every sample of a random signal is independent of all other samples. Such a signal, which is completely unpredictable, is known as *white noise* and is used as a building block to simulate random signals with different types of dependence. To summarize, the fundamental characteristic of a random signal is the inability to precisely specify its values. In other words, a random signal is not predictable, it never repeats itself, and we cannot find a mathematical formula that provides its values as a function of time. As a result, random signals can only be mathematically described by using the theory of stochastic processes.

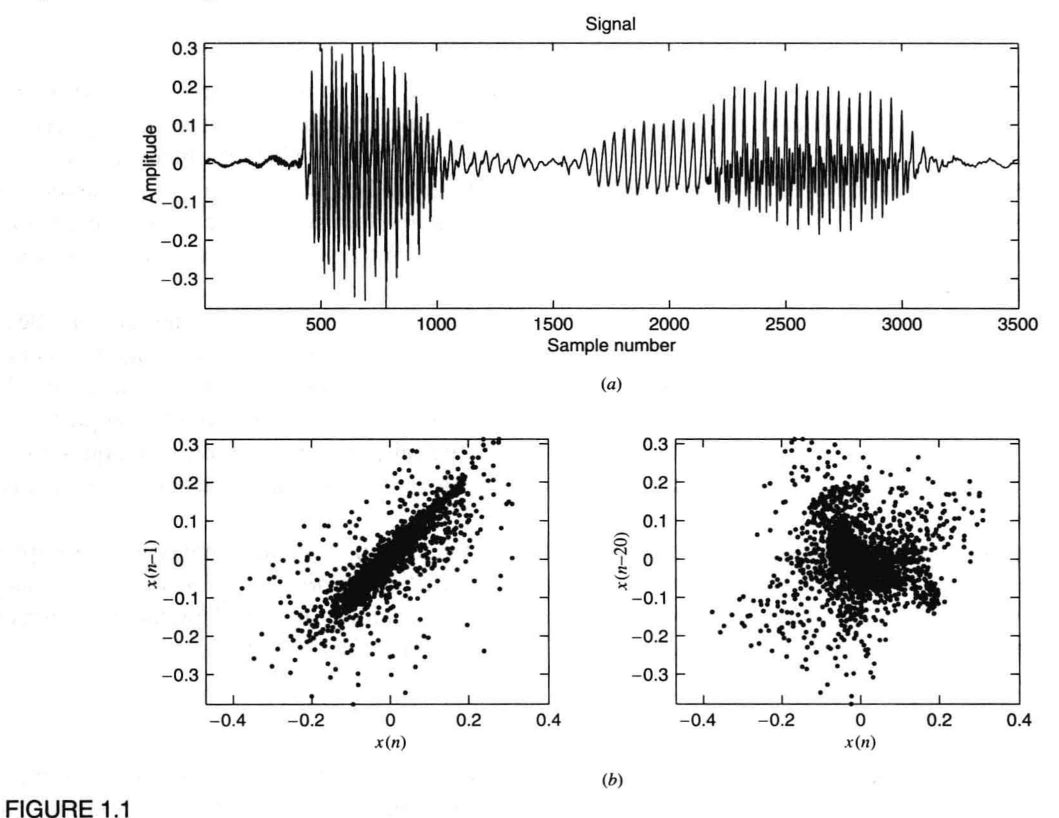


FIGURE 1.1

(*a*) The waveform for the speech signal "signal"; (*b*) two scatter plots for successive samples and samples separated by 20 sampling intervals.

This book provides an introduction to the fundamental theory and a broad selection of algorithms widely used for the processing of discrete-time random signals. Signal processing techniques, dependent on their main objective, can be classified as follows (see Figure 1.2):

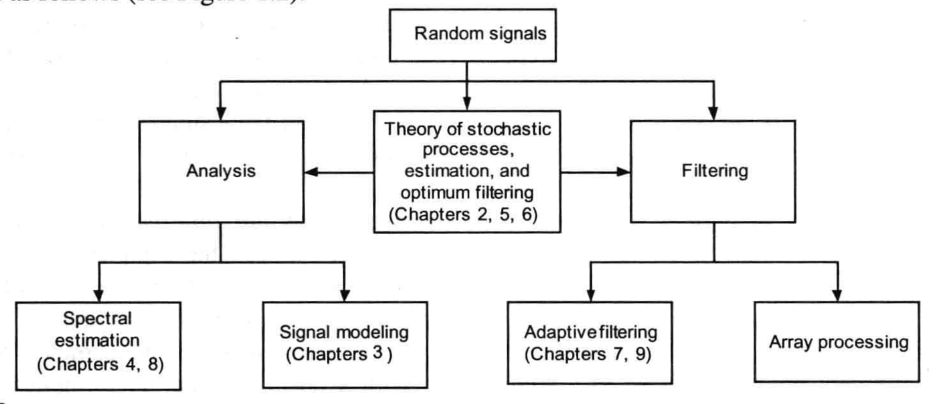


FIGURE 1.2

Classification of methods for the analysis and processing of random signals.

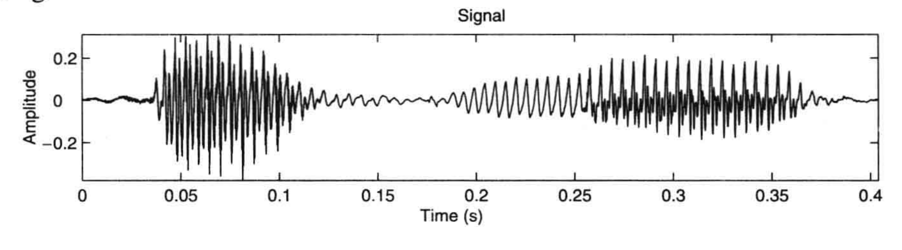
- **Signal analysis.** The primary goal is to extract useful information that can be used to understand the signal generation process or extract features that can be used for signal classification purposes. Most of the methods in this

area are treated under the disciplines of *spectral estimation* and *signal modeling*. Typical applications include detection and classification of radar and sonar targets, speech and speaker recognition, detection and classification of natural and artificial seismic events, event detection and classification in biological and financial signals, efficient signal representation for data compression, etc.

- **Signal filtering.** The main objective of signal filtering is to improve the quality of a signal according to an acceptable criterion of performance. Signal filtering can be subdivided into the areas of frequency selective filtering and adaptive filtering. Typical applications include noise and interference cancelation, echo cancelation, channel equalization, seismic deconvolution, active noise control, etc.

We conclude this section with some examples of signals occurring in practical applications. Although the description of these signals is far from complete, we provide sufficient information to illustrate their random nature and significance in signal processing applications.

Speech signals. Figure 1.3 shows the spectrogram and speech waveform corresponding to the utterance "signal." The spectrogram is a visual representation of the distribution of the signal energy as a function of time and frequency. We note that the speech signal has significant changes in both amplitude level and spectral content across time. The waveform contains segments of voiced (quasi-periodic) sounds, such as "e," and unvoiced or fricative (noiselike) sounds, such as "g."



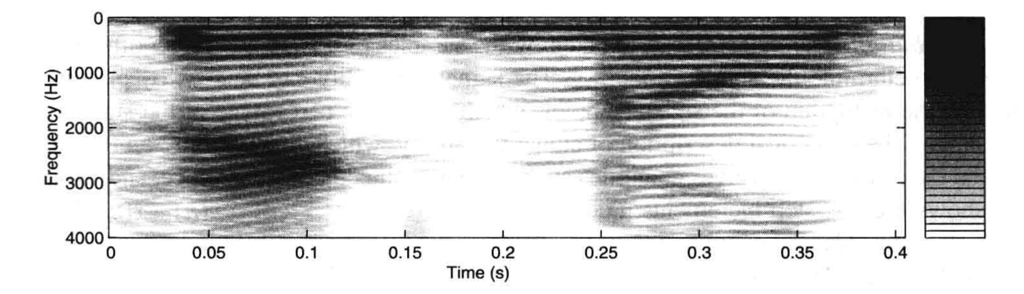


FIGURE 1.3
Spectrogram and acoustic waveform for the utterance "signal." The horizontal dark bands show the resonances of the vocal tract, which change as a function of time depending on the sound or phoneme being produced.

Speech production involves three processes: generation of the sound excitation, articulation by the vocal tract, and radiation from the lips and/or nostrils. If the excitation is a quasi-periodic train of air pressure pulses, produced by the vibration of the vocal cords, the result is a voiced sound. Unvoiced sounds are produced by first creating a constriction in the vocal tract, usually toward the mouth end. Then we generate turbulence by forcing air through the constriction at a sufficiently high velocity. The resulting excitation is a broadband noiselike waveform.

The spectrum of the excitation is shaped by the vocal tract tube, which has a frequency response that resembles the resonances of organ pipes or wind instruments. The resonant frequencies of the vocal tract tube are known as *formant frequencies*, or simply *formants*. Changing the shape of the vocal tract changes its frequency response and results in the generation of different sounds. Since the shape of the vocal tract changes slowly during continuous speech, we usually assume that it remains almost constant over intervals on the order of 10 ms. More details about speech signal generation and processing can be found in Rabiner and Schafer 1978; O'Shaughnessy 1987; and Rabiner and Juang 1993.

Other examples of the stochastic signals are Electrophysiological signals, geophysical signals, radar signals.

1.2 Spectral Estimation

The central objective of signal analysis is the development of quantitative techniques to study the properties of a

signal and the differences and similarities between two or more signals from the same or different sources. The major areas of random signal analysis are (1) statistical analysis of signal amplitude (i.e., the sample values); (2) analysis and modeling of the correlation among the samples of an individual signal; and (3) joint signal analysis (i.e., simultaneous analysis of two signals in order to investigate their interaction or interrelationships). These techniques are summarized in Figure 1.4. The prominent tool in signal analysis is spectral estimation, which is a generic term for a multitude of techniques used to estimate the distribution of energy or power of a signal from a set of observations. Spectral estimation is a very complicated process that requires a deep understanding of the underlying theory and a great deal of practical experience. Spectral analysis finds many applications in areas such as medical diagnosis, speech analysis, seismology and geophysics, radar and sonar, nondestructive fault detection, testing of physical theories, and evaluating the predictability of time series.

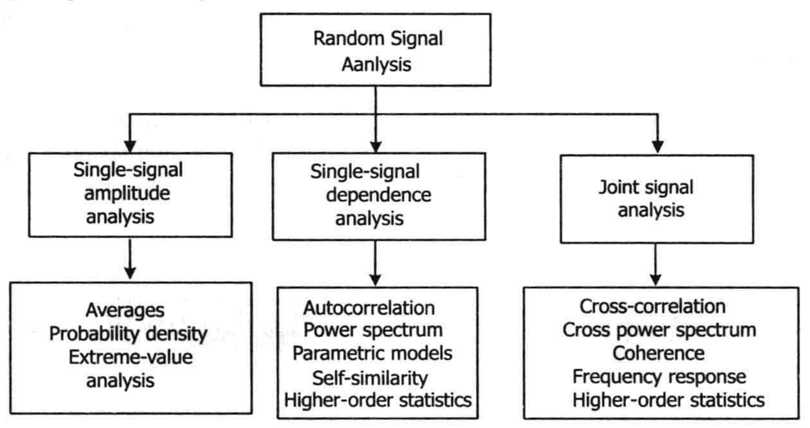


FIGURE 1.4
Summary of random signal analysis techniques.

Amplitude distribution. The range of values taken by the samples of a signal and how often the signal assumes these values together determine the signal variability. The signal variability can be seen by plotting the time series and is quantified by the histogram of the signal samples, which shows the percentage of the signal amplitude values within a certain range. The numerical description of signal variability, which depends only on the value of the signal samples and not on their ordering, involves quantities such as mean value, median, variance, and dynamic range.

Figure 1.5 shows the one-step increments, that is, the first difference $x_d(n)=x(n)-x(n-1)$ whereas Figure 1.6 shows their histograms. Careful examination of the shape of the histogram curves indicates that the second signal jumps quite frequently between consecutive samples with large steps. In other words, the probability of large increments is significant, as exemplified by the fat tails of the histogram in Figure 1.6(*b*). The knowledge of the probability of extreme values is essential in the design of detection systems for digital communications, military surveillance using infrared and radar sensors, and intensive care monitoring. In general, the shape of the histogram, or more precisely the probability density, is very important in applications such as signal coding and event detection. Although many practical signals follow a Gaussian distribution, many other signals of practical interest have distributions that are non-Gaussian. For example, speech signals have a probability density that can be reasonably approximated by a gamma distribution (Rabiner and Schafer 1978).

The significance of the Gaussian distribution in signal processing stems from the following facts. First, many physical signals can be described by Gaussian processes. Second, the central limit theorem states that any process that is the result of the combination of many elementary processes will tend, under quite general conditions, to be Gaussian. Finally, linear systems preserve the Gaussianity of their input signals. To understand the last two statements, consider N independent random quantities $x_1, x_2, \ldots, x_N$ with the same probability density $p(x)$ and pose the following question: When does the probability distribution $p_N(x)$ of their sum $x=x_1+x_2+\cdots+x_N$ have the same shape (within a scale factor) as the distribution $p(x)$ of the individual quantities? The standard answer is that $p(x)$ should be Gaussian, because the sum of N Gaussian random variables is again a Gaussian, but with variance equal to N times that of the individual signals. However, if we allow for distributions with infinite variance, additional solutions are possible. The resulting probability distributions, known as *stable* or *Levy distributions*, have infinite variance and are characterized by a thin main lobe and fat tails, resembling the shape of the histogram in Figure 1.6(*b*). Interestingly enough, the Gaussian distribution is a stable distribution with finite variance (actually the only one). Because Gaussian and stable non-Gaussian distributions are invariant under linear signal processing

operations, they are very important in signal processing.

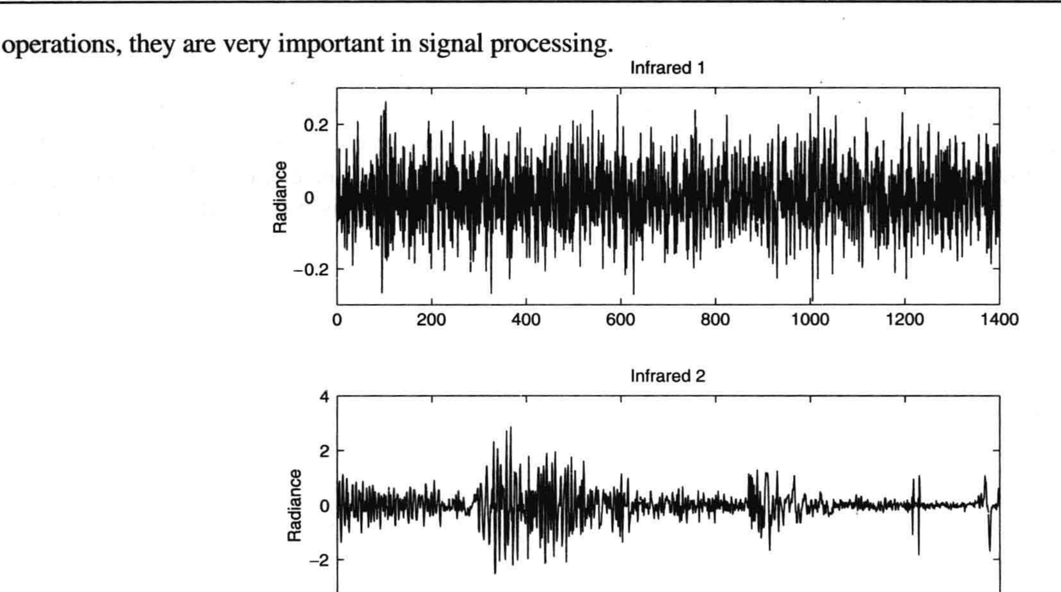
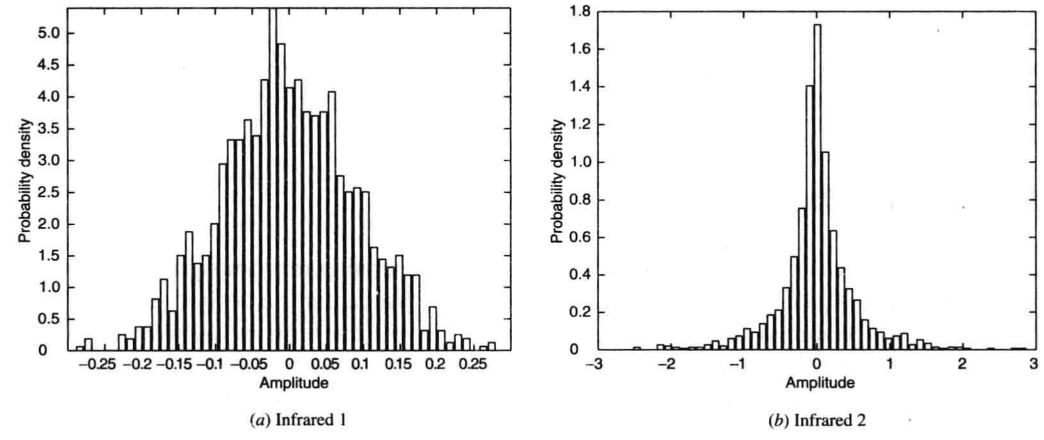


FIGURE 1.5
One-step-increment time series for the infrared data.



(a) Infrared 1 (b) Infrared 2

FIGURE 1.6
Histograms for the infrared increment signals.

Correlation and spectral analysis. Although scatter plots (see Figure 1.1) illustrate nicely the existence of correlation, to obtain quantitative information about the correlation structure of a time series $x(n)$ with zero mean value, we use the *empirical normalized autocorrelation sequence*

$$\hat{\rho}(l) = \frac{\sum_{n=l}^{N-1} x(n)x^*(n-l)}{\sum_{n=0}^{N-1} |x(n)|^2} \tag{1.2.1}$$

which is an estimate of the theoretical normalized autocorrelation sequence. For lag $l=0$, the sequence is perfectly correlated with itself and we get the maximum value of 1. If the sequence does not change significantly from sample to sample, the correlation of the sequence with its shifted copies, though diminished, is still close to 1. Usually, the correlation decreases as the lag increases because distant samples become less and less dependent. Note that reordering the samples of a time series changes its autocorrelation but not its histogram.

We say that signals whose empirical autocorrelation decays fast, such as an exponential, have short-memory or short-range dependence. If the empirical autocorrelation decays very slowly, as a hyperbolic function does, we say

that the signal has long-memory or long-range dependence. Furthermore, we shall see in the next section that effective modeling of time series with short or long memory requires different types of models.

The spectral density function shows the distribution of signal power or energy as a function of frequency (see Figure 1.7). The autocorrelation and the spectral density of a signal form a Fourier transform pair and hence contain the same information. However, they present this information in different forms, and one can reveal information that cannot be easily extracted from the other. It is fair to say that the spectral density is more widely used than the autocorrelation.

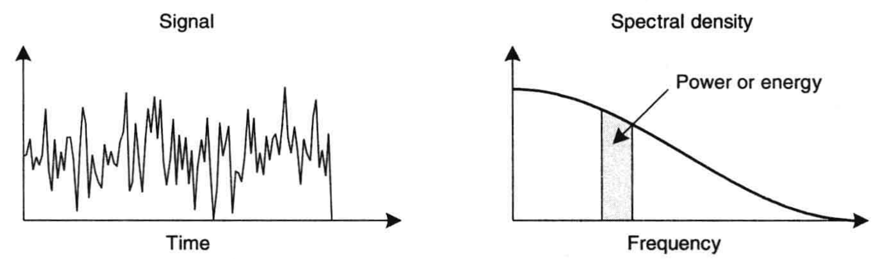


FIGURE 1.7
Illustration of the concept of power or energy spectral density function of a random signal.

Joint signal analysis. In many applications, we are interested in the relationship between two different random signals. There are two cases of interest. In the first case, the two signals are of the same or similar nature, and we want to ascertain and describe the similarity or interaction between them.

In the second case, we may have reason to believe that there is a *causal relationship* between the two signals. For example, one signal may be the input to a system and the other signal the output. The task in this case is to find an accurate description of the system, that is, a description that allows accurate estimation of future values of the output from the input. This process is known as *system modeling* or *identification* and has many practical applications, including understanding the operation of a system in order to improve the design of new systems or to achieve better control of existing systems.

1.3 Signal Modeling

In many theoretical and practical applications, we are interested in generating random signals with certain properties or obtaining an efficient representation of real-world random signals that captures a desired set of their characteristics (e.g., correlation or spectral features) in the best possible way. We use the term *model* to refer to a mathematical description that provides an efficient representation of the "essential" properties of a signal.

For example, a finite segment $\{x(n)\}_{n=0}^{N-1}$ of any signal can be approximated by a linear combination of constant $(\lambda_k = 1)$ or exponentially fading $(0 < \lambda_k < 1)$ sinusoids

$$x(n) \simeq \sum_{k=1}^{M} a_k \lambda_k^n \cos(\omega_k n + \varphi_k) \tag{1.3.1}$$

where $\{a_k, \lambda_k, \omega_k, \varphi_k\}_{k=1}^{M}$ are the model parameters. A good model should provide an accurate description of the signal with $4M \ll N$ parameters. From a practical viewpoint, we are most interested in *parametric models*, which assume a given functional form completely specified by a finite number of parameters. In contrast, *nonparametric models* do not put any restriction on the functional form or the number of model parameters.

If any of the model parameters in (1.3.1) is random, the result is a random signal. The most widely used model is given by

$$x(n) = \sum_{k=1}^{M} a_k \cos(\omega_k n + \varphi_k)$$

where the amplitudes $\{a_k\}_1^N$ and the frequencies $\{\omega_k\}_1^N$ are constants and the phases $\{\varphi_k\}_1^N$ are random. This model is known as the *harmonic process model* and has many theoretical and practical applications.

Suppose next that we are given a sequence $\omega(n)$ of independent and identically distributed observations. We can create a time series $x(n)$ with dependent observations, by linearly combining the values of $\omega(n)$ as

$$x(n) = \sum_{k=-\infty}^{\infty} h(k)\omega(n-k) \tag{1.3.2}$$

which results in the widely used *linear random signal model*. The model specified by the convolution summation (1.3.2) is clearly nonparametric because, in general, it depends on an infinite number of parameters. Furthermore, the model is a linear, time-invariant system with impulse response $h(k)$ that determines the *memory* of the model and, therefore, the dependence properties of the output $x(n)$. By properly choosing the weights $h(k)$, we can generate a time series with almost any type of dependence among its samples.

In practical applications, we are interested in linear parametric models. As we will see, parametric models exhibit a dependence imposed by their structure. However, if the number of parameters approaches the range of the dependence (in number of samples), the model can mimic any form of dependence. The list of desired features for a good model includes these: (1) the number of model parameters should be as small as possible (*parsimony*), (2) estimation of the model parameters from the data should be easy, and (3) the model parameters should have a physically meaningful interpretation.

If we can develop a successful parametric model for the behavior of a signal, then we can use the model for various applications:

1. To achieve a better understanding of the physical mechanism generating the signal (e.g., earth structure in the case of seismograms).
2. To track changes in the source of the signal and help identify their cause (e.g., EEG).
3. To synthesize artificial signals similar to the natural ones (e.g., speech, infrared backgrounds, natural scenes, data network traffic).
4. To extract parameters for pattern recognition applications (e.g., speech and character recognition).
5. To get an efficient representation of signals for data compression (e.g., speech, audio, and video coding).
6. To forecast future signal behavior (e.g., stock market indexes) (Pindyck and Rubinfeld 1998).

In practice, signal modeling involves the following steps: (1) selection of an appropriate model, (2) selection of the "right" number of parameters, (3) fitting of the model to the actual data, and (4) model testing to see if the model satisfies the user requirements for the particular application. As we shall see in Chapter 8 this process is very complicated and depends heavily on the understanding of the theoretical model properties (see Chapter 3), the amount of familiarity with the particular application, and the experience of the user.

Rational or Pole-Zero Models

Suppose that a given sample $x(n)$, at time n, can be approximated by the previous sample weighted by a coefficient a, that is, $x(n) \approx ax(n-1)$, where a is assumed constant over the signal segment to be modeled. To make the above relationship exact, we add an excitation term $\omega(n)$, resulting in

$$x(n) = ax(n-1) + \omega(n) \tag{1.3.3}$$

where $\omega(n)$ is an excitation sequence. Taking the z-transform of both sides, we have

$$X(z) = az^{-1}X(z) + W(z) \tag{1.3.4}$$

which results in the following system function:

$$H(z) = \frac{X(z)}{W(z)} = \frac{1}{1-az^{-1}} \tag{1.3.5}$$

By using the identity

$$H(z) = \frac{1}{1-az^{-1}} = 1 + az^{-1} + a^2z^{-2} + \cdots \qquad -1 < a < 1 \tag{1.3.6}$$

the single-parameter model in (1.3.3) can be expressed in the following nonparametric form

$$x(n) = \omega(n) + a\omega(n-1) + a^2\omega(n-2) + \cdots \tag{1.3.7}$$

which clearly indicates that the model generates a time series with *exponentially* decaying dependence.

A more general model can be obtained by including a linear combination of the P previous values of the signal and of the Q previous values of the excitation in (1.3.3), that is,

$$x(n) = \sum_{k=1}^{P}(-a_k)x(n-k) + \sum_{k=0}^{Q} d_k\omega(n-k) \tag{1.3.8}$$

The resulting system function

$$H(z)=\frac{X(z)}{W(z)}=\frac{\sum_{k=0}^{Q} d_k z^{-k}}{1+\sum_{k=1}^{P} a_k z^{-k}} \tag{1.3.9}$$

is *rational*, that is, a ratio of two polynomials in the variable z^{-1}, hence the term *rational models*. We will show in Chapter 3 that *any* rational model has a dependence structure or memory that decays exponentially with time. Because the roots of the numerator polynomial are known as *zeros* and the roots of the denominator polynomial as *poles*, these models are also known as *pole-zero models*. In the time-series analysis literature, these models are known as *autoregressive moving-average (ARMA)* models.

Modeling the vocal tract. An example of the application of the pole-zero model is for the characterization of the speech production system. Most generally, speech sounds are classified as either voiced or unvoiced. For both of these types of speech, the production is modeled by exciting a linear system, the vocal tract, with an excitation having a flat, that is, constant, spectrum. The vocal tract, in turn, is modeled by using a pole-zero system, with the poles modeling the vocal tract resonances and the zeros serving the purpose of dampening the spectral response between pole frequencies. In the case of voiced speech, the input to the vocal tract model is a quasi-periodic pulse waveform, whereas for unvoiced speech the source is modeled as random noise. The system model of the speech production process is shown in Figure 1.8. The parameters of this model are the voiced/unvoiced classification, the pitch period for voiced sounds, the gain parameter, and the coefficients $\{d_k\}$ and $\{a_k\}$ of the vocal tract filter (1.3.9). This model is widely used for low-bit-rate (less than 2.4 kbits/s) speech coding, synthetic speech generation, and extraction of features for speech and speaker recognition (Rabiner and Schafer 1978; Rabiner and Juang 1993;Furui 1989).

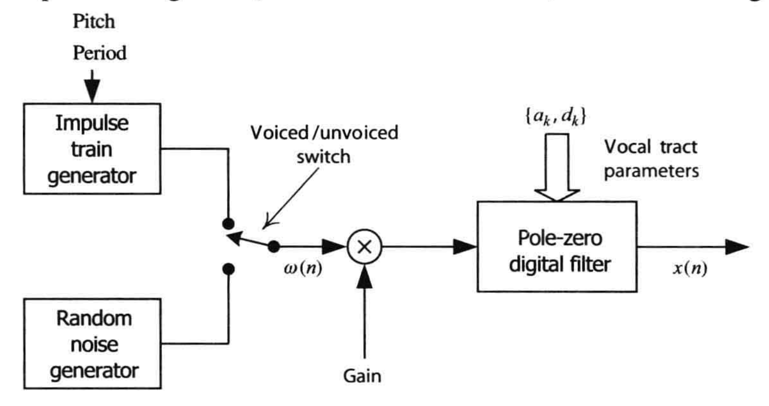


FIGURE 1.8
Speech synthesis system based on pole-zero modeling.

A classification of the various signal models described previously is given in Figure 1.9, which also provides information about the chapters of the book where these signals are discussed.

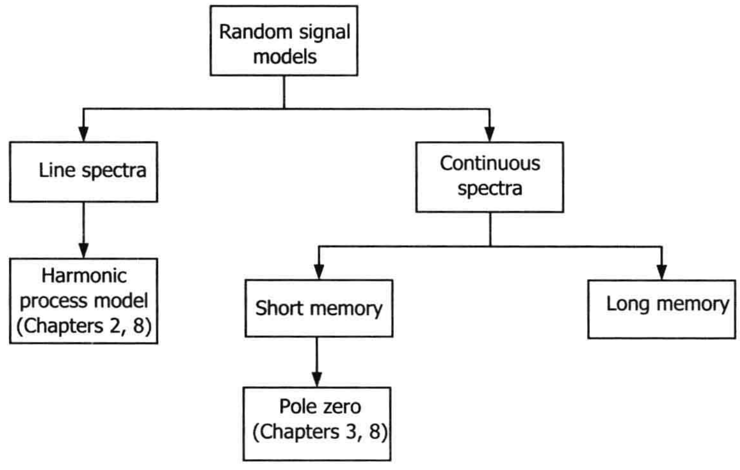


FIGURE 1.9
Classification of random signal models.

1.4 Adaptive Filtering

Conventional frequency-selective digital filters with *fixed* coefficients are designed to have a given frequency response chosen to alter the spectrum of the input signal in a desired manner. Their key features are as follows:

1. The filters are linear and time-invariant.
2. The design procedure uses the desired passband, transition bands, passband ripple, and stopband attenuation. We do *not* need to know the sample values of the signals to be processed.
3. Since the filters are frequency-selective, they work best when the various components of the input signal occupy nonoverlapping frequency bands. For example, it is easy to separate a signal and additive noise when their spectra do not overlap.
4. The filter coefficients are chosen during the design phase and are held constant during the normal operation of the filter.

However, there are many practical application problems that cannot be successfully solved by using fixed digital filters because either we do not have sufficient information to design a digital filter with fixed coefficients or the design criteria change during the normal operation of the filter. Most of these applications can be successfully solved by using special "smart" filters known collectively as *adaptive filters*. The distinguishing feature of adaptive filters is that they can modify their response to improve performance during operation without any intervention from the user.

1.4.1 Applications of Adaptive Filters

The best way to introduce the concept of adaptive filtering is by describing some typical application problems that can be effectively solved by using an adaptive filter. The applications of adaptive filters can be sorted for convenience into four classes: (1) system identification, (2) system inversion, (3) signal prediction, and (4) multisensor interference cancelation (see Figure 1.14 and Table 1.1). We next describe each class of applications and provide a typical example for each case.

TABLE 1.1

Classification of adaptive filtering applications.

Application class	Examples
System identification	Echo cancelation
	Adaptive control
	Channel modeling
System inversion	Adaptive equalization
	Blind deconvolution
Signal prediction	Adaptive predictive coding
	Change detection
	Radio frequency interference cancelation
Multisensor interference cancelation	Acoustic noise control
	Adaptive beamforming

System Identification

This class of applications, known also as system modeling, is illustrated in Figure 1.10 (*a*). The system to be modeled can be either real, as in control system applications, or some hypothetical signal transmission path (e.g., the echo path). The distinguishing characteristic of the system identification application is that the input of the adaptive filter is noise-free and the desired response is corrupted by additive noise that is uncorrelated with the input signal. Applications in this class include echo cancelation, channel modeling, and identification of systems for control applications (Gitlin et al. 1992; Ljung 1987; Åström and Wittenmark 1990). In control applications, the purpose of the adaptive filter is to estimate the parameters or the state of the system and then to use this information to design a controller. In signal processing applications, the goal is to obtain a good estimate of the desired response according to the adopted criterion of performance.

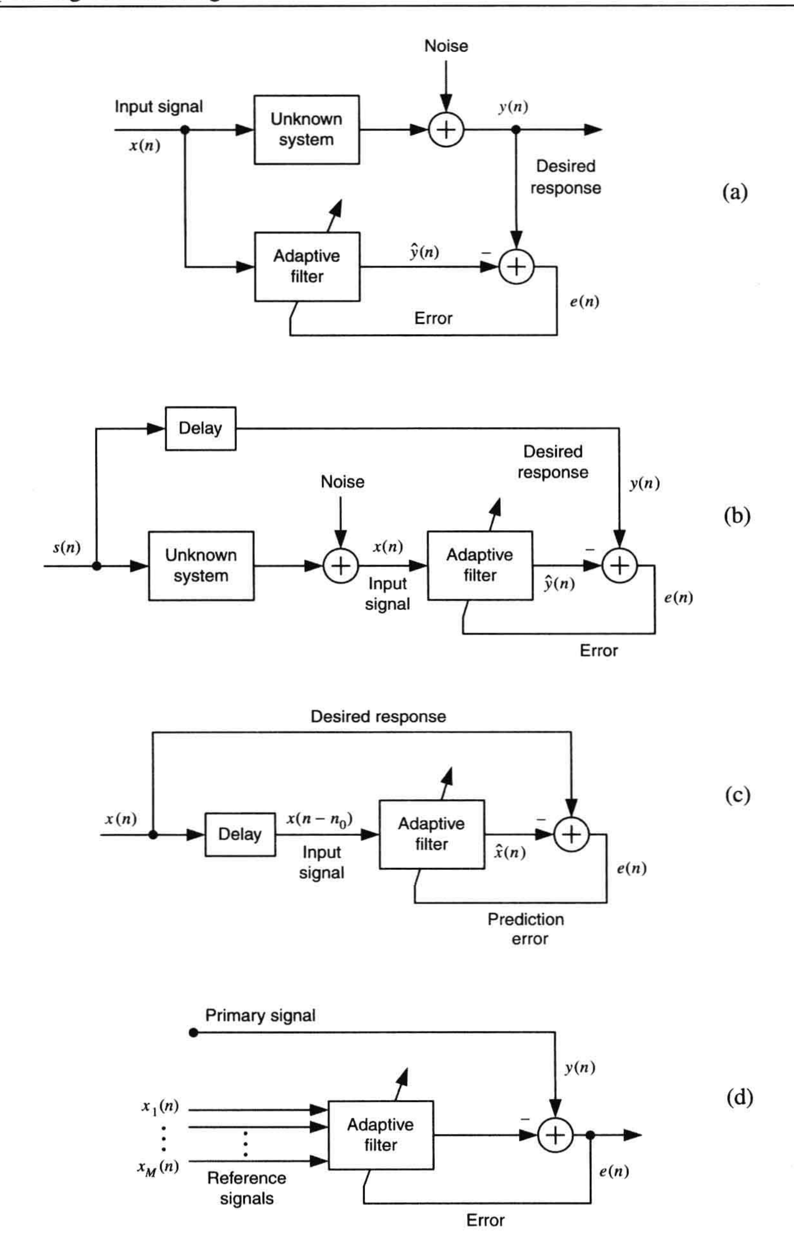


FIGURE 1.10

The four basic classes of adaptive filtering applications: (*a*) system identification, (*b*) system inversion, (*c*) signal prediction, and (*d*) multisensor interference cancelation.

Acoustic echo cancelation. Figure 1.11 shows a typical audio teleconferencing system that helps two groups of people, located at two different places, to communicate effectively. However, the performance of this system is degraded by the following effects: (1) The *reverberations* of the room result from the fact that the microphone picks up not only the speech coming from the talker but also reflections from the walls and furniture in the room. (2) *Echoes* are created by the acoustic coupling between the microphone and the loudspeaker located in the same room. Speech from room B not only is heard by the listener in room A but also is picked up by the microphone in room A, and unless it is prevented, will return as an echo to the speaker in room B.

Several methods to deal with acoustic echoes have been developed. However, the most effective technique to prevent or control echoes is adaptive echo cancelation. The basic idea is very simple: To cancel the echo, we generate a replica or pseudo-echo and then subract it from the real echo. To synthesize the echo replica, we pass the signal at the loudspeaker through a device designed to duplicate the reverberation and echo properties of the room (*echo path*), as is illustrated in Figure 1.12.

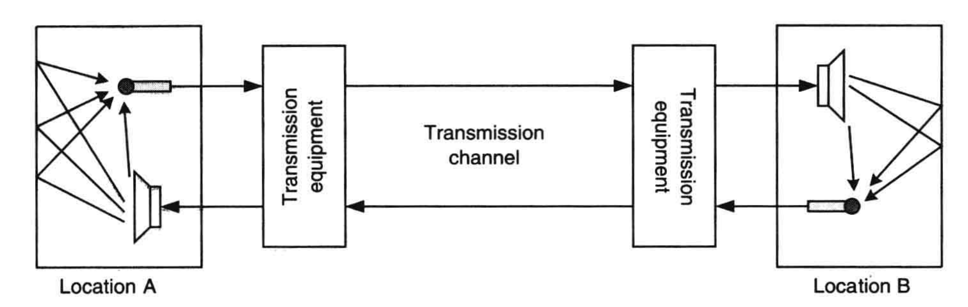


FIGURE 1.11
Typical teleconferencing system without echo control.

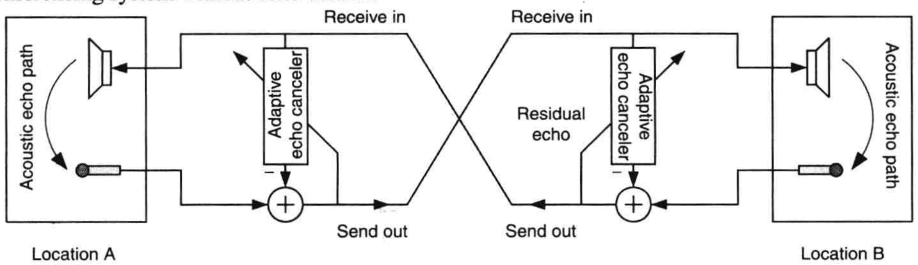


FIGURE 1.12
Principle of acoustic echo cancelation using an adaptive echo canceler.

In practice, there are two obstables to this approach. (1) The echo path is usually unknown before actual transmission begins and is quite complex to model. (2) The echo path is changing with time, since even the move of a talker alters the acoustic properties of the room. Therefore, we *cannot* design and use a *fixed* echo canceler with satisfactory performance for all possible connections. There are two possible ways around this problem:

1. Design a compromise *fixed* echo canceler based on some "average" echo path, assuming that we have sufficient information about the connections to be seen by the canceler.
2. Design an *adaptive echo canceler* that can "learn" the echo path when it is first turned on and afterward "tracks" its variations without any intervention from the designer. Since an adaptive canceler matches the echo patch for any given connection, it performs better than a fixed compromise canceler.

We stress that the main task of the canceler is to estimate the echo signal with sufficient accuracy; the estimation of the echo path is simply the means for achieving this goal. The performance of the canceler is measured by the attenuation of the echo. The adaptive echo canceler achieves this goal, by modifying its response, using the residual echo signal in an as-yet-unspecified way. More details about acoustic echo cancelation can be found in Gilloire et al. (1996).

System inversion

This class of applications, which is illustrated in Figure 1.10 (*b*), is also known as inverse system modeling. The goal of the adaptive filter is to estimate and apply the inverse of the system. Dependent on the application, the input of the adaptive filter may be corrupted by additive noise, and the desired response may not be available. The existence of the inverse system and its properties (e.g., causality and stability) creates additional complications. Typical applications include adaptive equalization (Gitlin et al. 1992), seismic deconvolution (Robinson 1984), and adaptive inverse control (Widrow and Walach 1994).

Channel equalization. To understand the basic principles of the channel equalization techniques, we consider a binary data communication system that transmits a band-limited analog pulse with amplitudes A (symbol 1) or $-A$ (symbol 0) every T_b s (see Figure 1.13). Here T_b is known as the *symbol interval* and $R_b = 1/T_b$ as the *baud rate*. As the signal propagates through the channel, it is delayed and attenuated in a frequency-dependent manner. Furthermore, it is corrupted by additive noise and other natural or man-made interferences. The goal of the receiver is to measure the amplitude of each arriving pulse and to determine which one of the two possible pulses has been sent. The received signal is sampled once per symbol interval after filtering, automatic gain control, and carrier removal. The sampling time is adjusted to coincide with the "center" of the received pulse. The shape of the pulse is chosen to attain the maximum rate at which the receiver can still distinguish the different pulses. To achieve this goal, we

usually choose a band-limited pulse that has periodic zero crossings every T_b s.

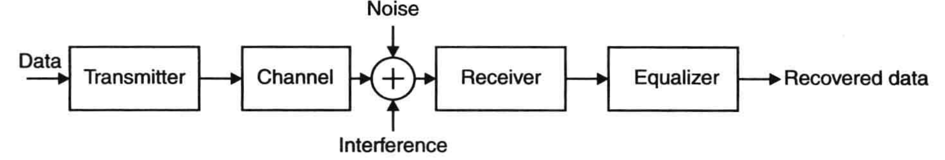


FIGURE 1.13

Simple model of a digital communications system.

If the periodic zero crossings of the pulse are preserved after transmission and reception, we can measure its amplitude without interference from overlapping adjacent pulses. However, channels that deviate from the ideal response (constant magnitude and linear phase) destroy the periodic zero-crossing property and the shape of the peak of the pulse. As a result, the tails of adjacent pulses interfere with the measurement of the current pulse and can lead to an incorrect decision. This type of degradation, which is known as *intersymbol interference (ISI)*, is illustrated in Figure 1.14.

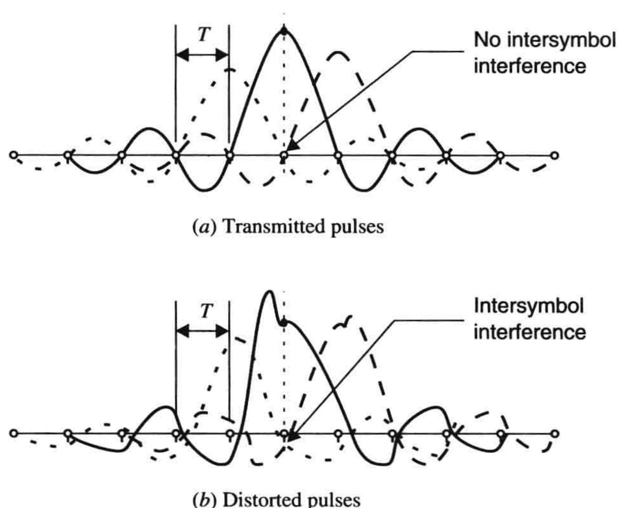


FIGURE 1.14

Pulse trains (*a*) without intersymbol interference and (*b*) with intersymbol interference.

We can compensate for the ISI distortion by using a linear filter called an *equalizer*. The goal of the equalizer is to restore the received pulse, as closely as possible, to its original shape. The equalizer transforms the channel to a near-ideal one if its response resembles the *inverse* of the channel. Since the channel is unknown and possibly time-varying, there are two ways to approach the problem: (1) Design a fixed compromise equalizer to obtain satisfactory performance over a broad range of channels, or (2) design an equalizer that can "learn" the inverse of the particular channel and then "track" its variation in real time.

The characteristics of the equalizer are adjusted by some algorithm that attempts to attain the best possible performance. The most appropriate criterion of performance for data transmission systems is the probability of symbol error. However it cannot be used for two reasons: (1) The "correct" symbol is unknown to the receiver (otherwise there would be no reason to communicate), and (2) the number of decisions (observations) needed to estimate the low probabilities of error is extremely large. Thus, practical equalizers assess their performance by using some function of the difference between the "correct" symbol and the output. The operation of practical equalizers involves two modes of operation, dependent on how we substitute for the unavailable correct symbol sequence. (1) A known *training sequence* is transmitted, and the equalizer attempts to improve its performance by comparing its output to a synchronized replica of the training sequence stored at the receiver. Usually this mode is used when the equalizer starts a transmission session. (2) At the end of the training session, when the equalizer starts making reliable decisions, we can replace the training sequence with the equalizer's own decisions.

Adaptive equalization is a mature technology that has had the greatest impact on digital communications systems, including voiceband, microwave and troposcatter radio, and cable TV modems (Qureshi 1985; Lee and Messerschmitt 1994; Gitlin et al. 1992; Bingham 1988; Treichler et al. 1996).

Signal prediction

In the next class of applications, the goal is to estimate the value $x(n_0)$ of a random signal by using a set of consecutive signal samples $\{x(n), n_1 \le n \le n_2\}$. There are three cases of interest: (1) forward prediction, when $n_0 > n_2$; (2) backward "prediction," when $n_0 < n_1$; and (3) smoothing or interpolation, when $n_1 < n_0 < n_2$. Clearly, in the last case the value at $n = n_0$ is not used in the computation of the estimate. The most widely used type is forward linear prediction or simply *linear prediction* [see Figure 1.10(*c*)], where the estimate is formed by using a linear combination of past samples (Makhoul 1975).

Linear predictive coding (LPC). The efficient storage and transmission of analog signals using digital systems requires the minimization of the number of bits necessary to represent the signal while maintaining the quality to an acceptable level according to a certain criterion of performance. The conversion of an analog (continuous-time, continuous-amplitude) signal to a digital (discrete-time, discrete-amplitude) signal involves two processes: sampling and quantization. Sampling converts a continuous-time signal to a discrete-time signal by measuring its amplitude at equidistant intervals of time. Quantization involves the representation of the measured continuous amplitude using a finite number of symbols and always creates some amount of distortion (quantization noise).

For a fixed number of bits, decreasing the dynamic range of the signal (and therefore the range of the quantizer) decreases the required quantization step and therefore the average quantization error power. Therefore, we can decrease the quantization noise by reducing the dynamic range or equivalently the variance of the signal. If the signal samples are significantly correlated, the variance of the difference between adjacent samples is smaller than the variance of the original signal. Thus, we can improve quality by quantizing this difference instead of the original signal. This idea is exploited by the linear prediction system shown in Figure 1.15. This system uses a linear predictor to form an estimate (prediction) $\hat{x}(n)$ of the present sample $x(n)$ as a linear combination of the M past samples, that is,

$$\hat{x}(n) = \sum_{k=1}^{M} a_k x(n-k) \tag{1.4.1}$$

The coefficients $\{a_k\}_1^M$ of the linear predictor are determined by exploiting the correlation between adjacent samples of the input signal with the objective of making the prediction error

$$e(n) = x(n) - \hat{x}(n) \tag{1.4.2}$$

as small as possible. If the prediction is good, the dynamic range of $e(n)$ should be smaller than the dynamic range of $x(n)$, resulting in a smaller quantization noise for the same number of bits or the same quantization noise with a smaller number of bits. The performance of the LPC system depends on the accuracy of the predictor. Since the statistical properties of the signal $x(n)$ are unknown and change with time, we *cannot* design an optimum fixed predictor. The established practical solution is to use an *adaptive* linear predictor that automatically adjusts its coefficients to compute a "good" prediction at each time instant. A detailed discussion of adaptive linear prediction and its application to audio, speech, and video signal coding is provided in Jayant and Noll (1984).

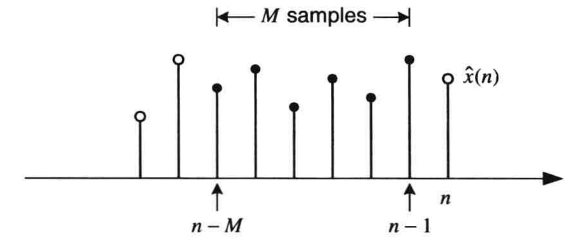


FIGURE 1.15
Illustration of the linear prediction of a signal $x(n)$ using a finite number of past samples.

Multisensor interference cancelation

The key feature of this class of applications is the use of multiple sensors to remove undesired interference and noise. Typically, a *primary* signal contains both the signal of interest and the interference. Other signals, known as *reference* signals, are available for the purposes of canceling the undesired interference [see Figure 1.10 (*d*)]. These reference signals are collected using other sensors in which the signal of interest is not present or is so weak that it can be ignored. The amount of correlation between the primary and reference signals is measured and used to form an

estimate of the interference in the primary signal, which is subsequently removed. Had the signal of interest been present in the reference signal(s), then this process would have resulted in the removal of the desired signal as well. Typical applications in which interference cancelation is employed include array processing for radar and communications, biomedical sensing systems, and active noise control (Widrow et al. 1975; Kuo and Morgan 1996).

Active noise control (ANC). The basic idea behind an ANC system is the cancelation of acoustic noise using destructive wave interference. To create destructive interference that cancels an acoustic noise wave (primary) at a point *P*, we can use a loudspeaker that creates, at the same point *P*, another wave (secondary) with the same frequency, the same amplitude, and 180° phase difference. Therefore, with appropriate control of the peaks and troughs of the secondary wave, we can produce zones of destructive interference (quietness). ANC systems using digital signal processing technology find applications in air-conditioning ducts, aircraft, cars, and magnetic resonance imaging (MRI) systems (Elliott and Nelson 1993; Kuo and Morgan 1996).

Figure 1.16 shows the key components of an adaptive ANC system described in Crawford et al. 1997. The task of the loudspeaker is to generate an acoustic wave that is an 180° phase-inverted version of the signal $y(t)$ when it arrives at the error microphone. In this case the error signal $e(t) = y(t) + \hat{y}(t) = 0$, and we create a "quiet zone" around the microphone. If the acoustic paths (1) from the noise source to the reference microphone (G_x), (2) from the noise source to the error microphone (G_y), (3) from the secondary loudspeaker to the reference microphone (H_x), and (4) from the secondary loudspeaker to the error microphone $(H_{\hat{y}})$ are linear, time-invariant, and known, we can design a linear filter H such that $e(n) = 0$. For example, if the effects of H_x and $H_{\hat{y}}$ are negligible, the filter H should invert G_x to obtain $v(t)$ and then replicate G_y to synthesize $\hat{y}(t) \simeq y(t)$. The quality of cancelation depends on the accuracy of these two modeling processes.

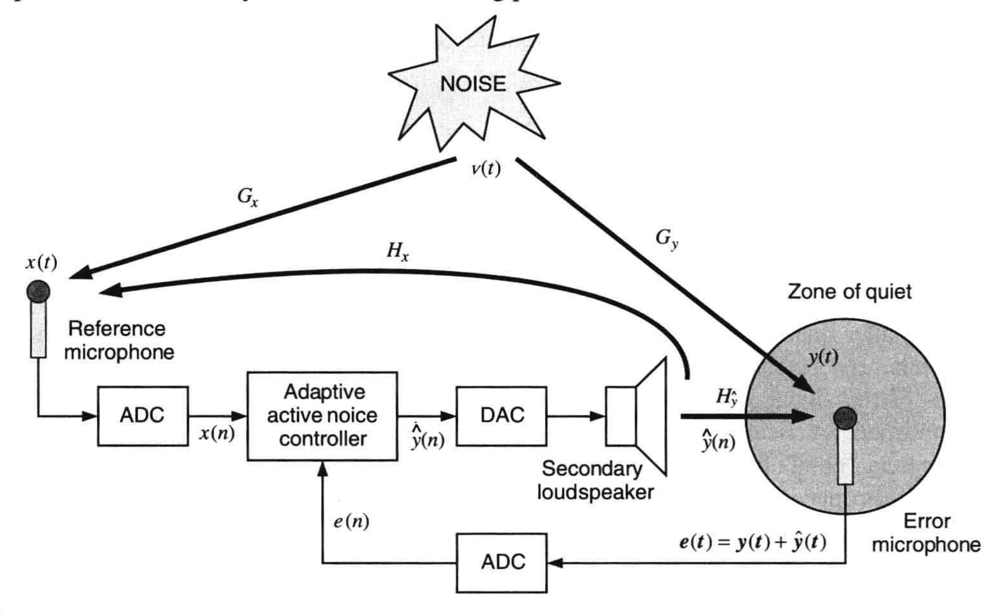


FIGURE 1.16
Block diagram of the basic components of an active noise control system.

In practice, the acoustic environment is unknown and time-varying. Therefore, we cannot design a fixed ANC filter with satisfactory performance. The only feasible solution is to use an adaptive filter with the capacity to identify and track the variation of the various acoustic paths and the spectral characteristics of the noise source in real time. The adaptive ANC filter adjusts its characteristics by trying to minimize the energy of the error signal $e(n)$. Adaptive ANC using digital signal processing technology is an active area of research, and despite several successes many problems remain to be solved before such systems find their way to more practical applications (Crawford et al. 1997).

1.4.2 *Features of Adaptive Filters*

Careful inspection of the applications discussed in the previous section indicates that every adaptive filter consists of the following three modules (see Figure 1.17).

1. **Filtering structure.** This module forms the output of the filter using measurements of the input signal or signals. The filtering structure is *linear* if the output is obtained as a linear combination of the input measurements; otherwise, it is said to be *nonlinear*. The structure is fixed by the designer, and its parameters are adjusted by the adaptive algorithm.
2. **Criterion of performance (COP).** The output of the adaptive filter and the desired response (when available) are processed by the COP module to assess its quality with respect to the requirements of the particular application.
3. **Adaptive algorithm.** The adaptive algorithm uses the value of the criterion of performance, or some function of it, and the measurements of the input and desired response (when available) to decide how to modify the parameters of the filter to improve its performance.

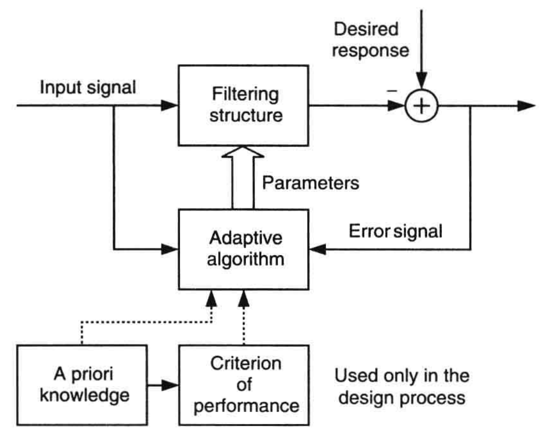


FIGURE 1.17
Basic elements of a general adaptive filter.

Every adaptive filtering application involves one or more input signals and a desired response signal that may or may not be accessible to the adaptive filter. We collectively refer to these relevant signals as the *signal operating environment* (*SOE*) of the adaptive filter. The design of any adaptive filter requires a great deal of *a priori information* about the SOE and a deep understanding of the particular application (Claasen and Mecklenbrauker 1985). This information is needed by the designer to choose the filtering structure and the criterion of performance and to design the adaptive algorithm. To be more specific, adaptive filters are designed for a specific type of input signal (speech, binary data, etc.), for specific types of interferences (additive white noise, sinusoidal signals, echoes of the input signals, etc.), and for specific types of signal transmission paths (e.g., linear time-invariant or time-varying). After the proper design decisions have been made, the only unknowns, when the adaptive filter starts its operation, are a set of parameters that are to be determined by the adaptive algorithm using signal measurements. Clearly, unreliable a priori information and/or incorrect assumptions about the SOE can lead to serious performance degradations or even unsuccessful adaptive filter applications.

If the characteristics of the relevant signals are constant, the goal of the adaptive filter is to find the parameters that give the best performance and then to stop the adjustment. However, when the characteristics of the relevant signals change with time, the adaptive filter should first find and then continuously readjust its parameters to track these changes.

A very influential factor in the design of adaptive algorithms is the availability of a desired response signal. We have seen that for certain applications, the desired response may not be available for use by the adaptive filter. In this book we focus on supervised adaptive filters that require the use of a desired response signal and we simply call them adaptive filters.

Suppose now that the relevant signals can be modeled by stochastic processes with known statistical properties. If we adopt the minimum mean square error as a criterion of performance, we can design, at least in principle, an optimum filter that provides the ultimate solution. From a theoretical point of view, the goal of the adaptive filter is to replicate the performance of the optimum filter without the benefit of knowing and using the *exact* statistical properties of the relevant signals. In this sense, the theory of optimum filters (see Chapters 5 and 6) is a prerequisite for the understanding, design, performance evaluation, and successful application of adaptive filters.

1.5 Organization of the Book

In this section we provide an overview of the main topics covered in the book so as to help the reader navigate through the material and understand the interdependence among the various chapters (see Figure 1.18).

In Chapter 2, we elaborate on certain topics that are crucial to developments in subsequent chapters. Reading this chapter is essential to familiarize the reader with notation and properties that are repeatedly used throughout the rest of the book. Chapter 4 presents the most practical methods for nonparametric estimation of correlation and spectral densities. The use of these techniques for exploratory investigation of the relevant signal characteristics before performing any modeling or adaptive filtering is invaluable.

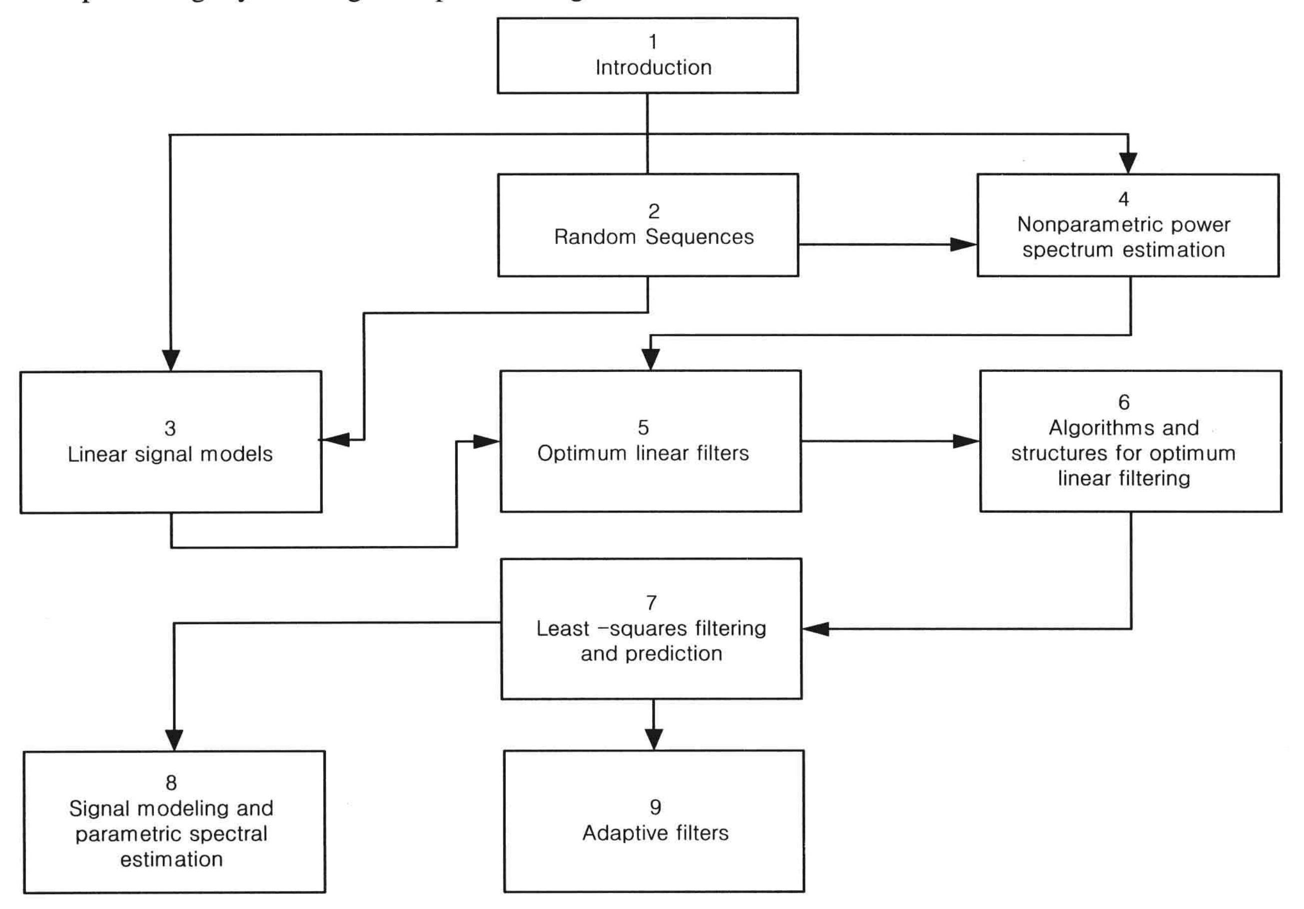


FIGURE 1.18
Flowchart organization of the book's chapters.

Chapters 3 and 5 provide a detailed study of the theoretical properties of signal models and optimum filters, assuming that the relevant signals can be modeled by stochastic processes with known statistical properties. In Chapter 6, we develop algorithms and structures for optimum filtering and signal modeling and prediction.

Chapter 7 introduces the general method of least squares and shows how to use it for the design of filters and predictors from actual signal observations. The statistical properties and the numerical computation of least-squares estimates are also discussed in detail.

Chapters 8 and 9 use the theoretical work in Chapters 3, 5 and 6 and the practical methods in Chapter 7 to develop, evaluate, and apply practical techniques for signal modeling, adaptive filtering.

CHAPTER 2

Random Sequences

Deterministic signals are the signals whose amplitude is uniquely specified by a mathematical formula or rule. However, there are many important examples of signals whose precise description (i.e., as deterministic signals) is extremely difficult, if not impossible. Although random signals are evolving in time in an unpredictable manner, their average properties can be often assumed to be deterministic; that is, they can be specified by explicit mathematical formulas. This is the key for the modeling of a random signal as a stochastic process.

Our aim in the subsequent discussions is to present some basic results from discrete-time stochastic processes that will be useful in the chapters that follow. We assume that most readers have some basic knowledge of these topics, and so parts of this chapter may be treated as a review exercise. However, some specific topics are developed in greater depth with a viewpoint that will serve as a foundation for the rest of the book. A more complete treatment can be found in Papoulis (1991), Helstrom (1992), and Stark and Woods (1994).

2.1 Discrete-Time Stochastic Processes

Many natural sequences can be characterized as random signals because we cannot determine their values precisely, that is, they are unpredictable. A natural mathematical framework for the description of these discrete-time random signals is provided by discrete-time stochastic processes.

To obtain a formal definition, consider an experiment with a finite or infinite number of unpredictable outcomes from a sample space $S=\{\zeta_1,\zeta_2,\cdots\}$, each occurring with a probability $\Pr\{\zeta_k\}, k=1,2,\cdots$. By some rule we assign to each element ζ_k of S a deterministic sequence $x(n,\zeta_k), -\infty<n<\infty$. The sample space S, the probabilities $\Pr\{\zeta_k\}$, and the sequences $x(n,\zeta_k), -\infty<n<\infty$, constitute a *discrete-time stochastic process* or random sequence. $x(n,\zeta), -\infty<n<\infty$, is a random sequence if for a fixed value n_0 of $n, x(n_0,\zeta)$ is a random variable.

The set of all possible sequences $\{x(n,\zeta)\}$ is called an *ensemble*, and each individual sequence $x(n,\zeta_k)$, corresponding to a specific value of $\zeta=\zeta_k$, is called a *realization* or a *sample sequence* of the ensemble.

There are four possible interpretations of $x(n,\zeta)$, depending on the character of n and ζ, as illustrated in Figure 2.1.

- $x(n,\zeta)$ is a random variable if n is *fixed* and ζ is a variable.
- $x(n,\zeta)$ is a sample sequence if ζ is *fixed* and n is a variable.
- $x(n,\zeta)$ is a number if both n and ζ are *fixed.*
- $x(n,\zeta)$ is a stochastic process if both n and ζ are variables.

A random sequence is also called a *time series* in the statistics literature. It is a sequence of random variables, or it can be thought of as an *infinite-dimensional* random vector. As with any collection of infinite objects, one has to be careful with the asymptotic (or convergence) properties of a random sequence. If n is a continuous variable taking values in R, then $x(n,\zeta)$ is an *uncountable* collection of random variables or an ensemble of waveforms. This ensemble is called a continuous-time stochastic process or a *random process*. Although these processes can be handled similarly to sequences, they are more difficult to deal with in a rigorous mathematical manner than sequences are. Furthermore, practical signal processing requires discrete-time signals. Hence in this book we consider random sequences rather than random waveforms.

Finally, in passing we note that the word *stochastic* is derived from the Greek word *stochasticos*, which means skillful in aiming or guessing. Hence, the terms *random process* and *stochastic process* will be used interchangeably throughout this book.

A deterministic signal is by definition exactly predictable. This assumes that there exists a certain functional relationship that completely describes the signal, even if this relationship is not available. The unpredictability of a random process is, in general, the combined result of two things. First, the selection of a single realization is based on the outcome of a random experiment. Second, no functional description is available for all

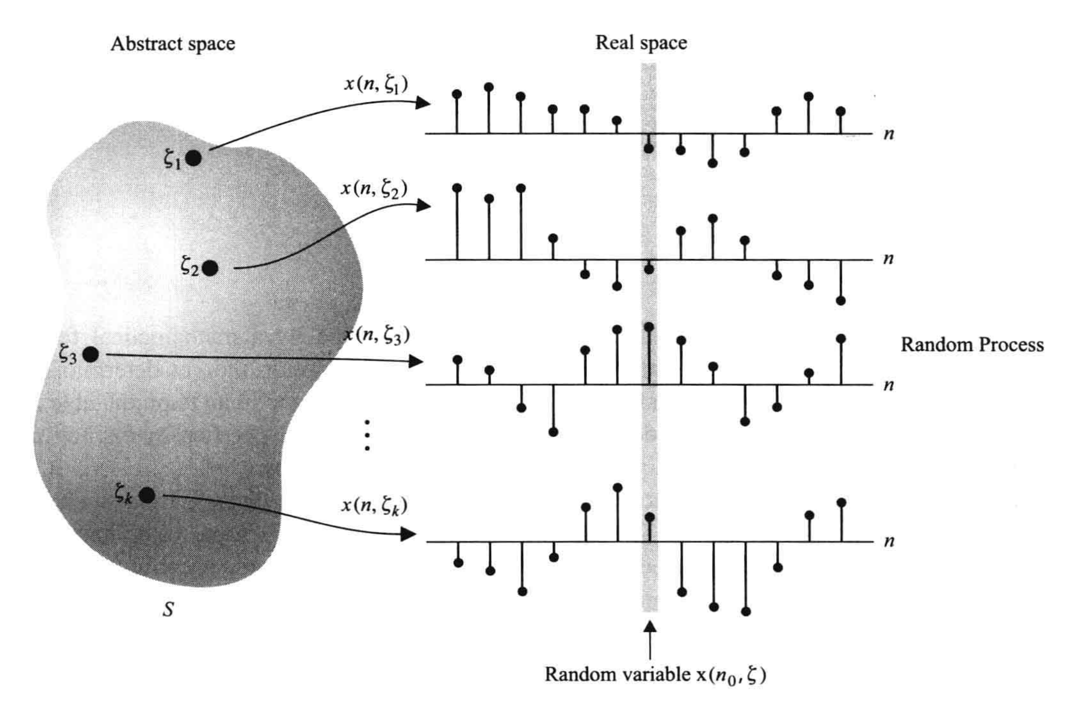


FIGURE 2.1
Graphical description of random sequences.

realizations of the ensemble. However, in some special cases, such a functional relationship is available. This means that after the occurrence of a specific realization, its future values can be predicted exactly from its past ones. If the future samples of any realization of a stochastic process can be predicted from the past ones, the process is called *predictable* or *deterministic*; otherwise, it is said to be a *regular process*. For example, the process $x(n,\zeta) = c$, where c is a random variable, is a predictable stochastic process because every realization is a discrete-time signal with constant amplitude. In practice, we most often deal with regular stochastic processes.

The simplest description of any random signal is provided by an amplitude-versus-time plot. Inspection of this plot provides qualitative information about some significant features of the signal that are useful in many applications. These features include, among others, the following:

1. The frequency of occurrence of various signal amplitudes, described by the probability distribution of samples.
2. The degree of dependence between two signal samples, described by the correlation between them.
3. The existence of "cycles" or quasi-periodic patterns, obtained from the signal power spectrum (which will be described in Section 2.1.6).
4. Indications of variability in the mean, variance, probability density, or spectral content.

The first feature above, the amplitude distribution, is obtained by plotting the histogram, which is an estimate of the first-order probability density of the underlying stochastic process. The probability density indicates waveform features such as "spikiness" and boundedness. Its form is crucial in the design of reliable estimators, quantizers, and event detectors.

The dependence between two signal samples (which are random variables) is given theoretically by the autocorrelation sequence and is quantified in practice by the empirical correlation, which is an estimate of the autocorrelation sequence of the underlying process. It affects the rate of amplitude change from sample to sample.

Cycles in the data are related to sharp peaks in the power spectrum or periodicity in the autocorrelation. Although the power spectrum and the autocorrelation contain the same information, they present it in different fashions.

Variability in a given quantity (e.g., variance) can be studied by evaluating this quantity for segments that can be assumed locally stationary and then analyzing the segment-to-segment variation. Such short-term descriptions should be distinguished from long-term ones, where the whole signal is analyzed as a single segment.

All the above features, to a lesser or greater extent, are interrelated. Therefore, it is impossible to point out exactly the effect of each one upon the visual appearance of the signal. However, a lot of insight can be gained by introducing the concepts of signal variability and signal memory, which are discussed in Sections 2.1.5 and 2.2.3

respectively.

2.1.1 Description Using Probability Functions

From Figure 2.1, it is clear that at $n=n_0$, $x(n_0,\zeta)$ is a random variable that requires a first-order probability function, say cdf $F_x(x;n_0)$, for its description. Similarly, $x(n_1,\zeta)$ and $x(n_2,\zeta)$ are joint random variables at instances n_1 and n_2, respectively, requiring a joint cdf $F_x(x_1,x_2;n_1,n_2)$. Stochastic processes contain infinitely many such random variables. Hence they are completely described, in a statistical sense, if their k th-order distribution function

$$F_x(x_1,\cdots,x_k;n_1,\cdots,n_k)=\Pr\{x(n_1)\le x_1,\cdots,x(n_k)\le x_k\} \tag{2.1.1}$$

is known for every value of $k\ge 1$ and for all instances $n_1,n_2,\ldots,n_k$. The k th-order pdf is given by

$$f_x(x_1,\cdots,x_k;n_1,\cdots,n_k)\triangleq\frac{\partial^{2k}F_x(x_1,\cdots,x_K;n_1,\cdots,n_k)}{\partial x_{R1}\cdots\partial x_{Ik}} \qquad k\ge 1 \tag{2.1.2}$$

Clearly, the probabilistic description requires a lot of information that is difficult to obtain in practice except for simple stochastic processes. However, many (but not all) properties of a stochastic process can be described in terms of *averages* associated with its first- and second-order densities.

For simplicity, in the rest of the book, we will use a compact notation $x(n)$ to represent either a random process $x(n,\zeta)$ or a single realization $x(n)$, which is a member of the ensemble. Thus we will drop the variable ζ from all notations involving random variables, vectors, or processes. We believe that this will not cause any confusion and that the exact meaning will be clear from the context. Also the random process $x(n)$ is assumed to be complex-valued unless explicitly specified as real-valued.

2.1.2 Second-Order Statistical Description

The second-order statistic of $x(n)$ at time n is specified by its *mean value* $\mu_x(n)$ and its *variance* $\sigma_x^2(n)$, defined by

$$\mu_x(n)=E\{x(n)\}=E\{x_R(n)+jx_I(n)\} \tag{2.1.3}$$

and

$$\sigma_x^2(n)=E\{|x(n)-\mu_x(n)|^2\}=E\{|x(n)|^2\}-|\mu_x(n)|^2 \tag{2.1.4}$$

respectively. Note that both $\mu_x(n)$ and $\sigma_x(n)$ are, in general, deterministic sequences.

The second-order statistics of $x(n)$ at two different times n_1 and n_2 are given by the two-dimensional autocorrelation (or autocovariance) sequences. The *autocorrelation sequence* of a discrete-time random process is defined as the joint moment of the random variables $x(n_1)$ and $x(n_2)$, that is,

$$r_{xx}(n_1,n_2)=E\{x(n_1)x^*(n_2)\} \tag{2.1.5}$$

It provides a measure of the dependence between values of the process at two different times. In this sense, it also provides information about the time variation of the process. The *autocovariance* sequence of $x(n)$ is defined by

$$\begin{aligned}\gamma_{xx}(n_1,n_2)&=E\{[x(n_1)-\mu_x(n_1)][x(n_2)-\mu_x(n_2)]^*\}\\&=r_{xx}(n_1,n_2)-\mu_x(n_1)\mu_x^*(n_2)\end{aligned} \tag{2.1.6}$$

We will use notations such as $\gamma_x(n_1,n_2), r_x(n_1,n_2), \gamma(n_1,n_2)$, or $r(n_1,n_2)$ when there is no confusion as to which signal we are referring. Note that, in general, the second-order statistics are defined on a two-dimensional grid of integers.

The statistical relation between two stochastic processes $x(n)$ and $y(n)$ that are jointly distributed (i.e., they are defined on the same sample space S) can be described by their *cross-correlation* and *cross-covariance* functions, defined by

$$r_{xy}(n_1,n_2) = E\{x(n_1)y^*(n_2)\} \tag{2.1.7}$$

and

$$\begin{aligned}\gamma_{xy}(n_1,n_2) &= E\{[x(n_1)-\mu_x(n_1)][y(n_2)-\mu_y(n_2)]^*\} \\ &= r_{xy}(n_1,n_2)-\mu_x(n_1)\mu_y^*(n_2)\end{aligned} \tag{2.1.8}$$

The *normalized cross-correlation* of two random processes $x(n)$ and $y(n)$ is defined by

$$\rho_{xy}(n_1,n_2) = \frac{\gamma_{xy}(n_1,n_2)}{\sigma_x(n_1)\sigma_y(n_2)} \tag{2.1.9}$$

Some definitions

We now describe some useful types of stochastic processes based on their statistical properties. A random process is said to be

- An *independent* process if

$$f_x(x_1,\cdots,x_k;n_1,\cdots,n_k) = f_1(x_1;n_1)\cdots f_k(x_k;n_k) \qquad \forall k, n_i, i=1,\cdots,k \tag{2.1.10}$$

  that is, $x(n)$ is a sequence of independent random variables. If all random variables have the same pdf $f(x)$ for all k, then $x(n)$ is called an IID (independent and identically distributed) random sequence.

- An *uncorrelated* process if $x(n)$ is a sequence of uncorrelated random variables, that is,

$$\gamma_x(n_1,n_2) = \begin{Bmatrix}\sigma_x^2(n_1) & n_1=n_2 \\ 0 & n_1\neq n_2\end{Bmatrix} = \sigma_x^2(n_1)\delta(n_1-n_2) \tag{2.1.11}$$

  Alternatively, we have

$$r_x(n_1,n_2) = \begin{cases}\sigma_x^2(n_1)+|\mu_x(n_1)|^2 & n_1=n_2 \\ \mu_x(n_1)\mu_x^*(n_2) & n_1\neq n_2\end{cases} \tag{2.1.12}$$

- An *orthogonal* process if it is a sequence of orthogonal random variables, that is,

$$r_x(n_1,n_2) = \begin{Bmatrix}\sigma_x^2(n_1)+|\mu_x(n_1)|^2 & n_1=n_2 \\ 0 & n_1\neq n_2\end{Bmatrix} = E\{|x(n_1)|^2\}\delta(n_1-n_2) \tag{2.1.13}$$

- An *independent increment* process if $\forall k>1$ and $\forall n_1<n_2<\cdots<n_k$, the *increments*

$$\{x(n_1)\},\{x(n_2)-x(n_1)\},\cdots,\{x(n_k)-x(n_{k-1})\}$$

  are jointly independent. For such sequences, the kth-order probability function can be constructed as products of the probability functions of its increments.

- A *wide-sense periodic* (*WSP*) process with period N if

$$\mu_x(n) = \mu_x(n+N) \qquad \forall n \tag{2.1.14}$$

  and

$$r_x(n_1,n_2) = r_x(n_1+N,n_2) = r_x(n_1,n_2+N) = r_x(n_1+N,n_2+N) \tag{2.1.15}$$

  Note that in the above definition, $\mu_x(n)$ is periodic in one dimension while $r_x(n_1,n_2)$ is periodic in two dimensions.

- A *wide-sense cyclostationary* process if there exists an integer N such that

$$\mu_x(n) = \mu_x(n+N) \qquad \forall n \tag{2.1.16}$$

and $$r_x(n_1,n_2)=r_x(n_1+N,n_2+N) \tag{2.1.17}$$

Note that in the above definition, $r_x(n_1,n_2)$ is *not* periodic in a two-dimensional sense. The correlation sequence is invariant to shift by N in *both* of its arguments.

• If all kth-order distributions of a stochastic process are jointly Gaussian, then it is called a Gaussian random sequence.

We can also extend some of these definitions to the case of two joint stochastic processes. The random processes $x(n)$ and $y(n)$ are said to be

• *Statistically independent* if for all values of n_1 and n_2

$$f_{xy}(x,y;n_1,n_2)=f_x(x;n_1)f_y(y;n_2) \tag{2.1.18}$$

• *Uncorrelated* if for every n_1 and n_2 $(n_1\neq n_2)$

$$\gamma_{xy}(n_1,n_2)=0 \quad \text{or} \quad r_{xy}(n_1,n_2)=\mu_x(n_1)\mu_y^*(n_2) \tag{2.1.19}$$

• *Orthogonal* if for every n_1 and n_2 $(n_1\neq n_2)$

$$r_{xy}(n_1,n_2)=0 \tag{2.1.20}$$

2.1.3 Stationarity

A random process $x(n)$ is called *stationary* if statistics determined for $x(n)$ are equal to those for $x(n+k)$, for every k. More specifically, we have the following definition.

DEFINITION 2.1 (STATIONARY OF ORDER N). A stochastic process $x(n)$ is called *stationary of order N* if

$$f_x(x_1,\cdots,x_N;n_1,\cdots,n_N)=f_x(x_1,\cdots,x_N;n_{1+k},\cdots,n_{N+k}) \tag{2.1.21}$$

for any value of k. If $x(n)$ is stationary for all orders $N=1,2,\cdots,$ it is said to be *strict-sense stationary* (*SSS*).

An IID sequence is SSS. However, SSS is more restrictive than necessary for most practical applications. A more relaxed form of stationarity, which is sufficient for practical problems, occurs when a random process is *stationary up to order* 2, and it is also known as wide-sense stationarity.

DEFINITION 2.2 (WIDE-SENSE STATIONARITY). A random signal $x(n)$ is called *wide-sense stationary* (*WSS*) if

1. Its mean is a constant independent of n, that is,

$$E\{x(n)\}=\mu_x \tag{2.1.22}$$

2. Its variance is also a constant independent of n, that is,

$$var[x(n)]=\sigma_x^2 \tag{2.1.23}$$

and

3. Its autocorrelation depends only on the distance $l=n_1-n_2$, called *lag*, that is,

$$r_x(n_1,n_2)=r_x(n_1-n_2)=r_x(l)=E\{x(n+l)x^*(n)\}=E\{x(n)x^*(n-l)\} \tag{2.1.24}$$

From (2.1.22), (2.1.24), and (2.1.5) it follows that the autocovariance of a WSS signal also depends only on $l=n_1-n_2$, that is,

$$\gamma_x(l)=r_x(l)-|\mu_x|^2 \tag{2.1.25}$$

EXAMPLE 2.1.1. Let $\omega(n)$ be a zero-mean, uncorrelated Gaussian random sequence with variance $\sigma^2(n)=1$.

a. Characterize the random sequence $\omega(n)$.

b. Define $x(n)=\omega(n)+\omega(n-1), -\infty<n<\infty$. Determine the mean and autocorrelation of $x(n)$. Also characterize $x(n)$.

Solution. Note that the variance of $\omega(n)$ is a constant.

a. Since uncorrelatedness implies independence for Gaussian random variables, $\omega(n)$ is an independent random sequence. Since its mean and variance are constants, it is at least stationary in the first order. Furthermore, from (2.1.12) or (2.1.13) we have

$$r_\omega(n_1,n_2)=\sigma^2\delta(n_1-n_2)=\delta(n_1-n_2)$$

Hence $\omega(n)$ is also a WSS random process.

b. The mean of $x(n)$ is zero for all n since $\omega(n)$ is a zero-mean process. Consider

$$\begin{aligned} r_x(n_1,n_2) &= E\{x(n_1)x(n_2)\} \\ &= E\{[\omega(n_1)+\omega(n_1-1)][\omega(n_2)+\omega(n_2-1)]\} \\ &= r_\omega(n_1,n_2)+r_\omega(n_1,n_2-1)+r_\omega(n_1-1,n_2) \\ &\quad +r_\omega(n_1-1,n_2-1) \\ &= \sigma^2\delta(n_1-n_2)+\sigma^2\delta(n_1-n_2+1) \\ &\quad +\sigma^2\delta(n_1-1-n_2)+\sigma^2\delta(n_1-1-n_2+1) \\ &= 2\delta(n_1-n_2)+\delta(n_1-n_2+1)+\delta(n_1-n_2-1) \end{aligned}$$

Clearly, $r_x(n_1,n_2)$ is a function of n_1-n_2. Hence

$$r_x(l)=2\delta(l)+\delta(l+1)+\delta(l-1)$$

Therefore, $x(n)$ is a WSS sequence. However, it is not an independent random sequence since both $x(n)$ and $x(n+1)$ depend on $\omega(n)$.

EXAMPLE 2.1.2. (WIENER PROCESS). Toss a fair coin at each $n, -\infty < n < \infty$. Let

$$\omega(n)=\begin{cases} +S & \text{if heads is outcome} \quad \Pr(\text{H})=1 \\ -S & \text{if tails is outcome} \quad \Pr(\text{T})=1 \end{cases}$$

where S is a step size. Clearly, $\omega(n)$ is an independent random process with

$$E\{\omega(n)\}=0$$

and

$$E\{\omega^2(n)\}=\sigma_\omega^2=S^2\left(\tfrac{1}{2}\right)+S^2\left(\tfrac{1}{2}\right)=S^2$$

Define a new random process $x(n)$, $n\geq 1$, as

$$\begin{aligned} x(1) &= \omega(1) \\ x(2) &= x(1)+\omega(2)=\omega(1)+\omega(2) \\ &\vdots \\ x(n) &= x(n-1)+\omega(n)=\sum_{i=1}^{n}\omega(i) \end{aligned}$$

Note that $x(n)$ is a running sum of independent steps or increments; thus it is an independent increment process. Such a sequence is called a *discrete Wiener process* or *random walk*. We can easily see that

$$E\{x(n)\}=E\left\{\sum_{i=1}^{n}\omega(i)\right\}=0$$

and

$$\begin{aligned} E\{x^2(n)\} &= E\left\{\sum_{i=1}^{n}\omega(i)\sum_{k=1}^{n}\omega(k)\right\}=E\left\{\sum_{i=1}^{n}\sum_{k=1}^{n}\omega(i)\omega(k)\right\} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{n}E\{\omega(i)\omega(k)\}=\sum_{i=1}^{n}E\{\omega^2(i)\}=nS^2 \end{aligned}$$

Therefore, random walk is a nonstationary(or *evolutionary*) process with zero mean and variance that grows with n, the number of steps taken.

It should be stressed at this point that although any strict-sense stationary signal is wide-sense stationary, the inverse is not always true, except if the signal is Gaussian.

Two random signals $x(n)$ and $y(n)$ are called *jointly wide-sense stationary* if each is wide-sense stationary and their cross-correlation depends only on $l=n_1-n_2$

$$r_{xy}(l)=E\{x(n)y^*(n-l)\}=r_{xy}(l)-\mu_x\mu_y^* \tag{2.1.26}$$

Note that as a consequence of wide-sense stationarity the two-dimensional correlation and covariance sequences become one-dimensional sequences. This is a very important result that ultimately allows for a nice spectral description of stationary random processes.

Properties of autocorrelation sequences

The autocorrelation sequence of a stationary process has many important properties (which also apply to

autocovariance sequences, but we will discuss mostly correlation sequences). Vector versions of these properties are discussed extensively in Section 2.2.4, and their proofs are explored in the problems.

PROPERTY 2.1.1. The average power of a WSS process $x(n)$ satisfies

$$r_x(0) = \sigma_x^2 + |\mu_x|^2 \geq 0 \tag{2.1.27}$$

and

$$r_x(0) \geq |r_x(l)| \quad \text{for all } l \tag{2.1.28}$$

Proof. See Problem 2.14and Property 2.1.6.

This property implies that the correlation attains its maximum value at zero lag and this value is nonnegative. The quantity $|\mu_x|^2$ is referred to as the *average dc power*, and the quantity $\sigma_x^2 = \gamma_x(0)$ is referred to as the *average ac power* of the random sequence. The quantity $r_x(0)$ then is the *total* average power of $x(n)$.

PROPERTY 2.1.2. The autocorrelation sequence $r_x(l)$ is a conjugate symmetric function of lag *l*, that is,

$$r_x^*(-l) = r_x(l) \tag{2.1.29}$$

Proof. It follows from Definition 2.2 and from (2.1.24).

PROPERTY 2.1.3. The autocorrelation sequence $r_x(l)$ is nonnegative definite; that is, for any $M > 0$ and any α_k, α_m

$$\sum_{k=1}^{M}\sum_{m=1}^{M} \alpha_k r_x(k-m)\alpha_m^* \geq 0 \tag{2.1.30}$$

This is a necessary and sufficient condition for a sequence $r_x(l)$ to be the autocorrelation sequence of a random sequence.
Proof. See Problem 2.15.

Since in this book we exclusively deal with wide-sense stationary processes, we will use the term *stationary* to mean wide-sense stationary. The properties of autocorrelation and cross-correlation sequences of jointly stationary processes, $x(n)$ and $y(n)$, are summarized in Table 2.1.

Although SSS and WSS forms are widely used in practice, there are processes with different forms of stationarity. Consider the following example.

EXAMPLE 2.1.3. Let $x(n)$ be a real-valued random process generated by the system

$$x(n) = \alpha x(n-1) + \omega(n) \qquad n \geq 0 \qquad x(-1) = 0 \tag{2.1.31}$$

where $\omega(n)$ is a stationary random process with mean μ_ω and $r_\omega(l) = \sigma_\omega^2 \delta(l)$. The process $x(n)$ generated using (2.1.31) is known as a *first-order autoregressive*, or AR(1), process,[1] and the process $\omega(n)$ is known as a *white noise* process (defined in Section 2.1.6). Determine the mean $\mu_x(n)$ of $x(n)$ and comment on its stationarity.

Solution. To compute the mean of $x(n)$, we express it as a function of $\{\omega(n), \omega(n-1), \cdots, \omega(0)\}$ as follows

$$\begin{aligned}
x(0) &= \alpha x(-1) + \omega(0) = \omega(0) \\
x(1) &= \alpha x(0) + \omega(1) = \alpha\omega(0) + \omega(1) \\
&\vdots \\
x(n) &= \alpha^n \omega(0) + \alpha^{n-1}\omega(1) + \cdots + \omega(n) = \sum_{k=0}^{n} \alpha^k \omega(n-k)
\end{aligned}$$

Hence the mean of $x(n)$ is given by

$$\mu_x(n) = E\left\{\sum_{k=0}^{n} \alpha^k \omega(n-k)\right\} = \mu_\omega\left(\sum_{k=0}^{n} \alpha^k\right) = \begin{cases} (1-\alpha^{n+1})/(1-\alpha)\mu_\omega & \alpha \neq 1 \\ (n+1)\mu_\omega & \alpha = 1 \end{cases}$$

Clearly, the mean of $x(n)$ depends on *n*, and hence it is nonstationary. However, if we assume that $|\alpha| < 1$ (which implies that the system is BIBO stable), then as $n \to \infty$, we obtain

$$\mu_x(n) = \mu_\omega \frac{1-\alpha^{n+1}}{1-\alpha} \xrightarrow[n\to\infty]{} \frac{\mu_x}{1-\alpha}$$

---

[1] Note that from (2.1.3), $x(n-1)$ completely determines the distribution for $x(n)$, and $x(n)$ completely determines the distribution for $x(n+1)$, and so on. If

$$f_{x(n)|x(n-1)\cdots}(x_n \mid x_{n-1}\cdots) = f_{x(n)|x(n-1)}(x_n \mid x_{n-1})$$

then the process is termed a *Markov process*.

Thus $x(n)$ approaches first-order stationarity for large n. Similar analysis for the autocorrelation of $x(n)$ shows that $x(n)$ approaches wide-sense stationarity for large n (see Problem 2.23).

The above example illustrates a form of stationarity called *asymptotic* stationarity. A stochastic process $x(n)$ is *asymptotically stationary* if the statistics of random variables $x(n)$ and $x(n+k)$ become stationary as $k \to \infty$. When LTI systems are driven by zero-mean uncorrelated-component random processes, the output process becomes asymptotically stationary in the *steady state*. Another useful form of stationarity is given by stationary increments. If the increments $\{x(n)-x(n-k)\}$ of a process $x(n)$ form a stationary process for every k, we say that $x(n)$ is a process with *stationary increments*. Such processes can be used to model data in various practical applications.

The simplest way, to examine in practice if a real-world signal is stationary, is to investigate the physical mechanism that produces the signal. If this mechanism is time-invariant, then the signal is stationary. In case it is impossible to draw a conclusion based on physical considerations, we should rely on statistical methods (Bendat and Piersol 1986; Priestley 1981). Note that stationarity in practice means that a random signal has statistical properties that do not change over the time interval we observe the signal. For evolutionary signals the statistical properties change continuously with time. An example of a highly nonstationary random signal is the signals associated with the vibrations induced in space vehicles during launch and reentry. However, there is a kind of random signal whose statistical properties change slowly with time. Such signals, which are stationary over short periods, are called *locally stationary* signals. Many signals of great practical interest, such as speech, EEG, and ECG, belong to this family of signals.

Finally, we note that general techniques for the analysis of nonstationary signals do not exist. Thus only special methods that apply to specific types of nonstationary signals can be developed. Many such methods remove the nonstationary component of the signal, leaving behind another component that can be analyzed as stationary (Bendat and Piersol 1986; Priestley 1981).

2.1.4 Ergodicity

A stochastic process consists of the ensemble and a probability law. If this information is available, the statistical properties of the process can be determined in a quite straightforward manner. However, in the real world, we have access to only a limited number (usually one) of realizations of the process. The question that arises then is, Can we infer the statistical characteristics of the process from a single realization?

This is possible for the class of random processes that are called *ergodic* processes. Roughly speaking, ergodicity implies that all the statistical information can be obtained from any single representative member of the ensemble.

Time averages

All the statistical averages that we have defined up to this point are known as *ensemble averages* because they are obtained by "freezing" the time variable and averaging over the ensemble (see Fig. 2.1). Averages of this type are formally defined by using the expectation operator $E\{\,\}$. Ensemble averaging is not used frequently in practice, because it is impractical to obtain the number of realizations needed for an accurate estimate. Thus the need for a different kind of average, based on only one realization, naturally arises. Obviously such an average can be obtained only by time averaging.

The *time average* of a quantity, related to a discrete-time random signal, is defined as

$$<(\cdot)> \triangleq \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} (\cdot) \tag{2.1.32}$$

Note that, owing to its dependence on a single realization, any time average is itself a random variable. The time average is taken over all time because all realizations of a random process exist for all time; that is, they are power signals.

For every ensemble average we can define a corresponding time average. The following time averages are of special interest:

$$\begin{aligned}
\text{Mean value} &= \langle x(n) \rangle \\
\text{Mean square} &= \langle |x(n)|^2 \rangle \\
\text{Variance} &= \langle |x(n) - \langle x(n) \rangle|^2 \rangle \\
\text{Autocorrelation} &= \langle x(n)x^*(n-l) \rangle \\
\text{Autocovariance} &= \langle [x(n) - \langle x(n) \rangle][x(n-l) - \langle x(n) \rangle]^* \rangle \\
\text{Cross-correlation} &= \langle x(n)y^*(n-l) \rangle \\
\text{Cross-covariance} &= \langle [x(n) - \langle x(n) \rangle][y(n-l) - \langle y(n) \rangle]^* \rangle
\end{aligned} \tag{2.1.33}$$

It is necessary to mention at this point the remarkable similarity between time averages and the correlation sequences for deterministic power signals. Although this is just a formal similarity, due to the fact that random signals are power signals, both quantities have the same properties. However, we should always keep in mind that although time averages are random variables (because they are functions of ζ), the corresponding quantities for deterministic power signals are fixed numbers or deterministic sequences.

Ergodic random processes

As we have already mentioned, in many practical applications only one realization of a random signal is available instead of the entire ensemble. In general, a single member of the ensemble does not provide information about the statistics of the process. However, if the process is stationary and ergodic, then all statistical information can be derived from only one typical realization of the process.

A random signal $x(n)$ is called *ergodic*[1] if its ensemble averages equal appropriate time averages. There are several degrees of ergodicity (Papoulis 1991). We will discuss two of them: ergodicity in the mean and ergodicity in correlation.

DEFINITION 2.3 (ERGODIC IN THE MEAN). A random process $x(n)$ is ergodic *in the mean* if

$$\langle x(n) \rangle = E\{x(n)\} \tag{2.1.34}$$

DEFINITION 2.4 (ERGODIC IN CORRELATION). A random process $x(n)$ is *ergodic in correlation* if

$$\langle x(n)x^*(n-l) \rangle = E\{x(n)x^*(n-l)\} \tag{2.1.35}$$

Note that since $\langle x(n) \rangle$ is constant and $\langle x(n)x^*(n-l) \rangle$ is a function of l, if $x(n)$ is ergodic in both the mean and correlation, then it is also WSS. Thus only stationary signals can be ergodic. On the other hand, WSS does not imply ergodicity of any kind. Fortunately, in practice almost all stationary processes are also ergodic, which is very useful for the estimation of their statistical properties. From now on we will use the term *ergodic* to mean both ergodicity in the mean and ergodicity in correlation.

DEFINITION 2.5 (JOINT ERGODICITY). Two random signals are called *jointly ergodic* if they are individually ergodic and in addition

$$\langle x(n)y^*(n-l) \rangle = E\{x(n)y^*(n-l)\} \tag{2.1.36}$$

A physical interpretation of ergodicity is that one realization of the random signal $x(n)$, as time n tends to infinity, takes on values with the same statistics as the value $x(n_1)$, corresponding to all samples of the ensemble members at a given time $n = n_1$.

In practice, it is of course impossible to use the time-average formulas introduced above, because only finite records of data are available. In this case, it is common practice to replace the operator (2.1.32) by the operator

$$\langle (\cdot) \rangle_N = \frac{1}{2N+1} \sum_{n=-N}^{N} (\cdot) \tag{2.1.37}$$

to obtain *estimates* of the true quantities. Our desire in such problems is to find estimates that become increasingly accurate (in a sense to be defined in Section 2.4) as the length $2N+1$ of the record of used data becomes larger.

[1] Strictly speaking, the form of ergodicity that we will use is called *mean-square ergodicity* since the underlying convergence of random variables is in the mean-square sense (Stark and Woods 1994). Therefore, equalities in the definitions are in the mean-square sense.

Finally, to summarize, we note that whereas stationarity ensures the time invariance of the statistics of a random signal, ergodicity implies that any statistics can be calculated either by averaging over all members of the ensemble at a fixed time or by time-averaging over any single representative member of the ensemble.

2.1.5 Random Signal Variability

If we consider a stationary random sequence $\omega(n)$ that is *IID* with zero mean, its key characteristics depend on its first-order density. Figure 2.2 shows the probability density functions and sample realizations for IID processes with uniform, Gaussian, and Cauchy probability distributions. In the case of the uniform distribution, the amplitude of the random variable is limited to a range, with values occurring outside this interval with zero probability. On the other hand, the Gaussian distribution does not have a finite interval of support, allowing for the possibility of any value. The same is true of the Cauchy distribution, but its characteristics are dramatically different from those of the Gaussian distribution. The center lobe of the density is much narrower while the tails that extend out to infinity are significantly higher. As a result, the realization of the Cauchy random process contains numerous spikes or extreme values while the remainder of the process is more compact about the mean. Although the Gaussian random process allows for the possibility of large values, the probability of their occurrence is so small that they are not found in realizations of the process.

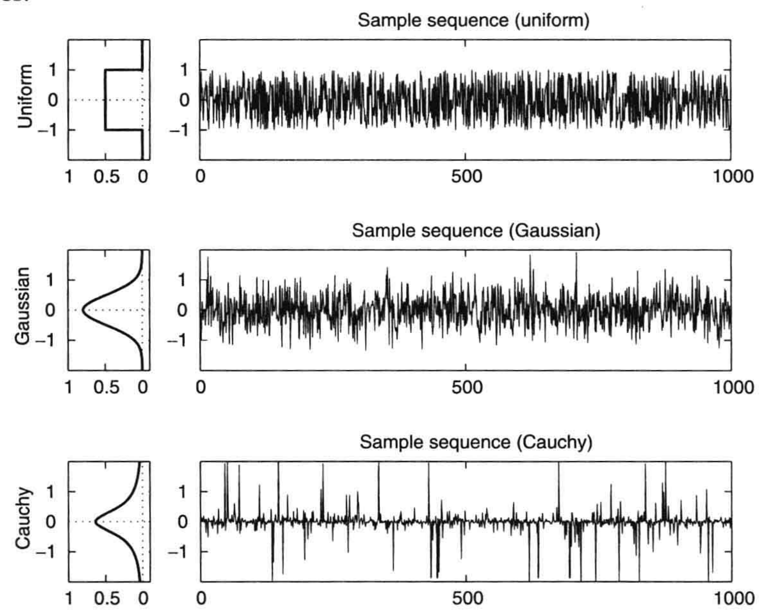


FIGURE 2.2

Probability density functions and sample realizations of an IID process with uniform, Gaussian, and Cauchy distributions.

The major difference between the Gaussian and Cauchy distributions lies in the area found under the tails of the density as it extends out to infinity. This characteristic is related to the *variability* of the process. The heavy tails, as found in the Cauchy distribution, result in an abundance of spikes in the process, a characteristic referred to as *high variability*. On the other hand, a distribution such as the Gaussian does not allow for extreme values and indicates *low variability*. The extent of the variability of a given distribution is determined by the heaviness of the tails. Distributions with heavy tails are called *long-tailed* distributions and have been used extensively as models of impulsive random processes.

DEFINITION 2.6. A distribution is called *long-tailed* if its tails decay hyperbolically or algebraically as

$$Pr\{|x(n)|\geq x\} \sim Cx^{-\alpha} \qquad \text{as } x\to\infty \tag{2.1.38}$$

where C is a constant and the variable α determines the rate of decay of the distribution.

By means of comparison, the Gaussian distribution has an exponential rate of decay. The implication of the algebraically decaying tail is that the process has infinite variance, that is,

$$\sigma_x^2 = E\{|x(n)|^2\} = \infty$$

and therefore lacks second-order moments. The lack of second-order moments means that, in addition to the variance,

the correlation functions of these processes do not exist. Since most signal processing algorithms are based on second-order moment theory, infinite variance has some extreme implications for the way in which such processes are treated.

In this book, we shall model high variability, and hence infinite variance, using the family of symmetric stable distributions. The reason is twofold: First, a linear combination of stable random variables is stable. Second, stable distributions appear as limits in central limit theorems. Stable distributions are characterized by a parameter $\alpha, 0<\alpha \leq 2$. They are Cauchy when $\alpha=1$ and Gaussian when $\alpha=2$. However, they have finite variance only when $\alpha=2$.

In practice, the type of data under consideration governs the variability of the modeling distribution. Random signals restricted to a certain interval, such as the phase of complex random signals, are well suited for uniform distributions. On the other hand, signals allowing for any possible value but generally confined to a region are better suited for Gaussian models. However, if a process contains spikes and therefore has high variability, it is best characterized by a long-tailed distribution such as the Cauchy distribution. Impulsive signals have been found in a variety of applications, such as communication channels, radar signals, and electronic circuit noise. In all cases, the variability of the process dictates the appropriate model.

2.1.6 *Frequency-Domain Description of Stationary Processes*

Discrete-time stationary random processes have correlation sequences that are functions of a single index. This leads to nice and powerful representations in both the frequency and the z-transform domains.

Power spectral density

The *power spectral density* (PSD, or more appropriately autoPSD) of a stationary stochastic process $x(n)$ is a Fourier transformation of its autocorrelation sequence $r_x(l)$. If $r_x(l)$ is periodic (which corresponds to a wide-sense periodic stochastic process) in *l*, then the DTFS can be used to obtain the PSD, which has the form of a *line spectrum*. If $r_x(l)$ is nonperiodic, the DTFT can be used, provided that $r_x(l)$ is absolutely summable. This means that the process $x(n)$ must be a zero-mean process. In general, a stochastic process can be a mixture of periodic and nonperiodic components.[2]

If we allow impulse functions in the DTFT to represent periodic (or almost periodic) sequences and non-zero-mean processes, then we can define the PSD as

$$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l) e^{-j\omega l} \tag{2.1.39}$$

where ω is the frequency in radians per sample. If the process $x(n)$ is a zero-mean nonperiodic process, then (2.1.39) is enough to determine the PSD. If $x(n)$ is periodic (including nonzero mean) or almost periodic, then the PSD is given by

$$R_x(e^{j\omega}) = \sum_i 2\pi A_i \delta(\omega - \omega_i) \tag{2.1.40}$$

where the A_i are amplitudes of $r_x(l)$ at frequencies ω_i. For discussion purposes we will assume that $x(n)$ is a zero-mean nonperiodic process. The autocorrelation $r_x(l)$ can be recovered from the PSD by using the inverse DTFT as

$$r_x(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) e^{j\omega l} \, d\omega \tag{2.1.41}$$

EXAMPLE 2.1.4. Determine the PSD of a zero-mean WSS process $x(n)$ with $r_x(l) = a^{|l|}, -1<a<1$.

Solution. From (2.1.39) we have

$$\begin{aligned} R_x(e^{j\omega}) &= \sum_{l=-\infty}^{\infty} a^{|l|} e^{-j\omega l} = \frac{1}{1-ae^{j\omega}} + \frac{1}{1-ae^{-j\omega}} - 1 \\ &= \frac{1-a^2}{1+a^2-2a\cos\omega} \qquad -1<a<1 \end{aligned} \tag{2.1.42}$$

[2]Periodic components are predictable processes as discussed before. However, some nonperiodic components can also be predictable. Hence nonperiodic components are not always regular processes.

which is a real-valued, even, and nonnegative function of ω.

Properties of the autoPSD. The power spectral density $R_x(e^{j\omega})$ has three key properties that follow from corresponding properties of the autocorrelation sequence and the DTFT.

PROPERTY 2.1.4. The autoPSD $R_x(e^{j\omega})$ is a real-valued periodic function of frequency with period 2π for any (real- or complex-valued) process $x(n)$. If $x(n)$ is real-valued, then $R_x(e^{j\omega})$ is also an even function of ω, that is,

$$R_x(e^{j\omega}) = R_x(e^{-j\omega}) \tag{2.1.43}$$

Proof. It follows from autocorrelation and DTFT properties.

PROPERTY 2.1.5. The autoPSD is nonnegative definite, that is,

$$R_x(e^{j\omega}) \geq 0 \tag{2.1.44}$$

Proof. This follows from the nonnegative definiteness of the autocorrelation sequence [see also discussions leading to (2.2.27)].

PROPERTY 2.1.6. The area under $R_x(e^{j\omega})$ is *nonnegative* and it equals the average power of $x(n)$ Indeed, from (2.1.41) it follows with $l=0$ that

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} R_x(e^{j\omega})\,d\omega = r_x(0) = E\{|x(n)|^2\} \geq 0 \tag{2.1.45}$$

Proof. It follows from Property 2.1.5.

White noise. A random sequence $\omega(n)$ is called a (second-order) *white noise process* with mean μ_ω and variance σ_ω^2, denoted by

$$\omega(n) \sim WN(\mu_\omega, \sigma_\omega^2) \tag{2.1.46}$$

if and only if $E\{\omega(n)\} = \mu_x$ and

$$r_\omega(l) = E\{\omega(n)\omega^*(n-l)\} = \sigma_\omega^2\delta(l) \tag{2.1.47}$$

which implies that

$$R_\omega(e^{j\omega}) = \sigma_\omega^2 \qquad -\pi \leq \omega \leq \pi \tag{2.1.48}$$

The term *white noise* is used to emphasize that all frequencies contribute the same amount of power, as in the case of white light, which is obtained by mixing all possible colors by the same amount. If, in addition, the pdf of $x(n)$ is Gaussian, then the process is called a (second-order) white Gaussian noise process, and it will be denoted by WGN($\mu_\omega, \sigma_\omega^2$).

If the random variables $\omega(n)$ are independently and identically distributed with mean μ_ω and variance σ_ω^2, then we shall write

$$\omega(n) \sim IID(\mu_\omega, \sigma_\omega^2) \tag{2.1.49}$$

This is sometimes referred to as a *strict* white noise.

We emphasize that the conditions of uncorrelatedness or independence do not put any restriction on the form of the probability density function of $\omega(n)$. Thus we can have an IID process with any type of probability distribution. Clearly, white noise is the simplest random process because it does not have any structure. However, we will see that it can be used as the basic building block for the construction of processes with more complicated dependence or correlation structures.

Harmonic processes. A *harmonic process* is defined by

$$x(n) = \sum_{k=1}^{M} A_k \cos(\omega_k n + \varphi_k) \qquad \omega_k \neq 0 \tag{2.1.50}$$

where $M, \{A_k\}_1^M$, and $\{\omega_k\}_1^M$ are constants, and $\{\varphi_k\}_1^M$ are pairwise independent random variables uniformly distributed in the interval $[0, 2\pi]$. It can be shown (see Problem 2.9) that $x(n)$ is a stationary process with mean

$$E\{x(n)\} = 0 \qquad \text{for all } n \tag{2.1.51}$$

and autocorrelation

$$r_x(l) = \frac{1}{2}\sum_{k=1}^{M} A_k^2 \cos \omega_k l \qquad -\infty < l < \infty \tag{2.1.52}$$

We note that $r_x(l)$ consists of a sum of "in-phase" cosines with the same frequencies as in $x(n)$.

If $\omega_k/(2\pi)$ are rational numbers, $r_x(l)$ is periodic and can be expanded as a Fourier series. These series coefficients provide the power spectrum $R_x(k)$ of $x(n)$. However, because $r_x(l)$ is a linear superposition of cosines, it always has a line spectrum with $2M$ lines of strength $A_k^2/4$ at frequencies $\pm\omega_k$. If $r_x(l)$ is periodic, then the lines are equidistant (i.e., harmonically related), hence the name *harmonic process*. If $\omega/(2\pi)$ is irrational, then $r_x(l)$ is almost periodic and can be treated in the frequency domain in almost the same fashion. Hence the power spectrum of a harmonic process is given by

$$R_x(e^{j\omega}) = \sum_{k=-M}^{M} 2\pi\left(\frac{A_k^2}{4}\right)\delta(\omega-\omega_k) = \sum_{k=-M}^{M} \frac{\pi}{2} A_k^2 \delta(\omega-\omega_k) \tag{2.1.53}$$

EXAMPLE 2.1.5. Consider the following harmonic process

$$x(n) = \cos(0.1\pi n + \varphi_1) + 2\sin(1.5n + \varphi_2)$$

where φ_1 and φ_2 are IID random variables uniformly distributed in the interval $[0, 2\pi]$. The first component of $x(n)$ is periodic with $\omega_1 = 0.1\pi$ and period equal to 20 while the second component is almost periodic with $\omega_2 = 1.5$. Thus the sequence $x(n)$ is almost periodic. A sample function realization of $x(n)$ is shown in Figure 2.3(*a*). The mean of $x(n)$ is

$$\mu_x(n) = E\{x(n)\} = E\{\cos(0.1\pi n + \varphi_1) + 2\sin(1.5n + \varphi_2)\} = 0$$

and the autocorrelation sequence (using mutual independence between φ_1 and φ_2) is

$$\begin{aligned} r_x(n_1,n_2) &= E\{x(n_1)x_2^*(n_2)\} \\ &= E\{\cos(0.1\pi n_1 + \varphi_1)\cos(0.1\pi n_2 + \varphi_1)\} \\ &\quad + E\{2\sin(1.5n_1 + \varphi_2)2\sin(1.5n_2 + \varphi_2)\} \\ &= \frac{1}{2}\cos[0.1\pi(n_1 - n_2)] + 2\cos[1.5(n_1 - n_2)] \end{aligned}$$

or

$$r_x(l) = \frac{1}{2}\cos 0.1\pi l + 2\cos 1.5l \qquad l = n_1 - n_2$$

Thus the line sepectrum $R_{\omega_k}^{(x)}$ is given by

$$R_{\omega_k}^{(x)} = \begin{cases} 1 & \omega_1 = -1.5 \\ \frac{1}{4} & \omega_2 = -0.1\pi \\ \frac{1}{4} & \omega_3 = 0.1\pi \\ 1 & \omega_4 = 1.5 \end{cases}$$

and the power spectrum $R_x(e^{j\omega})$ is given by

$$R_x(e^{j\omega}) = 2\pi\delta(\omega+1.5) + \frac{\pi}{2}\delta(\omega+0.1\pi) + \frac{\pi}{2}\delta(\omega-0.1\pi) + 2\pi\delta(\omega-1.5)$$

The line spectrum of $x(n)$ is shown in Figure 2.3(*b*) and the corresponding power spectrum in Figure 2.3(*c*).

The harmonic process is predictable because any given realization is a sinusoidal sequence with fixed amplitude, frequency, and phase. We stress that the independence of the phases is required to guarantee the stationarity of $x(n)$ in (2.1.50). The uniform distribution of the phases is necessary to make $x(n)$ a stationary process (see Problem 2.9). The harmonic process (2.1.50), in general, is non-Gaussian; however, it becomes Gaussian if the amplitudes A_k are random variables with a Rayleigh distribution (Porat 1994).

EXAMPLE 2.1.6. Consider a complex-valued process given by

$$x(n) = Ae^{j\omega_0 n} = |A|e^{j(\omega_0 n+\varphi)}$$

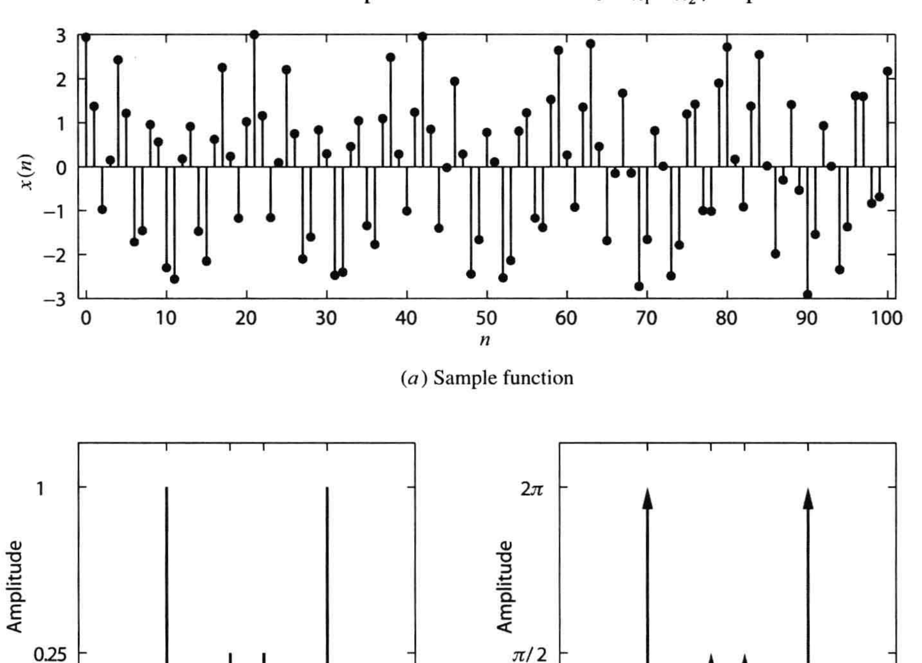
where A is a complex-valued random variable and ω_0 is constant. The mean of $x(n)$

$$E\{x(n)\} = E\{A\}e^{j\omega_0 n}$$

can be constant only if $E\{A\}=0$. If $|A|$ is constant and φ is uniformly distributed on $[0,2\pi]$, then we have $E\{A\} = |A|E\{e^{j\varphi}\} = 0$. In this case the autocorrelation is

$$r_x(n_1,n_2) = E\{Ae^{j(\omega_0 n_1+\varphi)}A^* e^{-j(\omega_0 n_2+\varphi)}\} = |A|^2 e^{j(n_1-n_2)\omega_0}$$

Since the mean is constant and the autocorrelation depends on the difference $l \triangleq n_1 - n_2$, the process is wide-sense stationary.



(a) Sample function

(b) Line spectrum

(c) Power spectrum

FIGURE 2.3
The time and frequency-domain description of the harmonic process in Example 2.1.5.

The above example can be generalized to harmonic processes of the form

$$x(n) = \sum_{k=1}^{M} A_k e^{j(\omega_k n+\varphi_k)} \tag{2.1.54}$$

where $M, \{A_k\}_1^M$, and $\{\omega_k\}_1^M$ are constants and $\{\varphi_k\}_1^M$ are pairwise independent random variables uniformly distributed in the interval $[0,2\pi]$. The autocorrelation sequence is

$$r_x(l) = \sum_{k=1}^{M} |A_k|^2 e^{j\omega_k l} \tag{2.1.55}$$

and the power spectrum consists of M impulses with amplitudes $2\pi|A_k|^2$ at frequencies ω_k. If the amplitudes $\{A_k\}_{k=1}^M$ are random variables, mutually independent of the random phases, the quantity $|A_k|^2$ is replaced by $E\{|A_k|^2\}$.

Cross-power spectral density

The cross-power spectral density of two zero-mean and jointly stationary stochastic processes provides a description of their statistical relations in the frequency domain and is defined as the DTFT of their cross-correlation, that is,

$$R_{xy}(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_{xy}(l)e^{-j\omega l} \tag{2.1.56}$$

The cross-correlation $r_{xy}(l)$ can be recovered by the inverse DTFT

$$r_{xy}(l) = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_{xy}(e^{j\omega})e^{j\omega l}\,d\omega \tag{2.1.57}$$

The cross-spectrum $R_{xy}(e^{j\omega})$ is, in general, a complex function of ω. From $r_{xy}(l) = r_{yx}^*(-l)$ it follows that

$$R_{xy}(e^{j\omega}) = R_{yx}^*(e^{j\omega}) \tag{2.1.58}$$

This implies that $R_{xy}(e^{j\omega})$ and $R_{yx}(e^{j\omega})$ have the same magnitude but opposite phase.

The normalized cross-spectrum

$$\mathcal{G}_{xy}(e^{j\omega}) \triangleq \frac{R_{xy}(e^{j\omega})}{\sqrt{R_x(e^{j\omega})}\sqrt{R_y(e^{j\omega})}} \tag{2.1.59}$$

is called the *coherence function*. Its squared magnitude

$$|\mathcal{G}_{xy}(e^{j\omega})|^2 = \frac{|R_{xy}(e^{j\omega})|^2}{R_x(\omega)R_y(e^{j\omega})} \tag{2.1.60}$$

is known as the *magnitude square coherence* (*MSC*) and can be thought of as a sort of correlation coefficient in the frequency domain. If $x(n)=y(n)$, then $\mathcal{G}_{xy}(e^{j\omega}) = 1$ (maximum correlation) whereas if $x(n)$ and $y(n)$ are uncorrelated, then $R_{xy}(l) = 0$ and hence $\mathcal{G}_{xy}(e^{j\omega}) = 0$. In other words, $0 \le |\mathcal{G}_{xy}(e^{j\omega})| \le 1$.

Complex spectral density functions

If the sequences $r_x(l)$ and $r_{xy}(l)$ are absolutely summable within a certain ring of the complex z plane, we can obtain their z-transforms

$$R_x(z) = \sum_{l=-\infty}^{\infty} r_x(l)z^{-l} \tag{2.1.61}$$

$$R_{xy}(z) = \sum_{l=-\infty}^{\infty} r_{xy}(l)z^{-l} \tag{2.1.62}$$

which are known as the *complex spectral density* and *complex cross-spectral density* functions, respectively. If the unit circle, defined by $z = e^{j\omega}$, is within the region of convergence of the above summations, then

$$R_x(e^{j\omega}) = R_x(z)|_{z=e^{j\omega}} \tag{2.1.63}$$

$$R_{xy}(e^{j\omega}) = R_{xy}(z)|_{z=e^{j\omega}} \tag{2.1.64}$$

The correlation and power spectral density properties of random sequences are summarized in Table 2.1.

EXAMPLE 2.1.7. Consider the random sequence given in Example 2.1.4 with autoPSD in (2.1.42)

$$R_x(e^{j\omega}) = \frac{1-a^2}{1+a^2-2a\cos\omega} \qquad |a|<1$$

Determine the complex autoPSD $R_x(z)$.

Solution. The complex autoPSD is given by $R_x(z) = R_x(e^{j\omega})|_{e^{j\omega}=z}$. Since

$$\cos\omega = \frac{e^{j\omega}+e^{-j\omega}}{2} = \left.\frac{z+z^{-1}}{2}\right|_{z=e^{j\omega}}$$

we obtain

$$R_x(z) = \frac{1-a^2}{1+a^2-2a\left(\frac{z+z^{-1}}{2}\right)} = \frac{(a-a^{-1})z^{-1}}{1-(a+a^{-1})z^{-1}+z^{-2}} \qquad |a|<|z|<\frac{1}{|a|}$$

Now the inverse z-transform of $R_x(z)$ determines the autocorrelation sequence $r_x(l)$, that is,

$$R_x(z) = \frac{(a-a^{-1})z^{-1}}{1-(a+a^{-1})z^{-1}+z^{-2}} = \frac{(a-a^{-1})z^{-1}}{(1-az^{-1})(1-a^{-1}z^{-1})}$$

$$= \frac{1}{(1-az^{-1})} - \frac{1}{(1-a^{-1}z^{-1})} \qquad |a|<|z|<|a|^{-1}$$

or

$$r_x(l) = a^l u(l) + (a^{-1})^l u(-l-1) = a^{|l|} \tag{2.1.65}$$

This approach can be used to determine autocorrelation sequences from autoPSD functions.

Table 2.1 provides a summary of correlation and spectral properties of stationary random sequences.

Table 2.1
Summary of correlation and spectral properties of stationary random sequences.

Definitions	
Mean value	$\mu_x = E\{x(n)\}$
Autocorrelation	$r_x(l) = E\{[x(n)x^*(n-l)\}$
Autocovariance	$\gamma_x(l) = E\{[x(n)-\mu_x][x(n-l)-\mu_x]^*\}$
Cross-correlation	$r_{xy}(l) = E\{x(n)y^*(n-l)\}$
Cross-covariance	$\gamma_{xy}(l) = E\{[x(n)-\mu_x][y(n-l)-\mu_y]^*\}$
Power spectral density	$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l}$
Cross-power spectral density	$R_{xy}(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_{xy}(l)e^{-j\omega l}$
Magnitude square coherence	$\|G_{xy}(e^{j\omega})\|^2 = \|R_{xy}(e^{j\omega})\|^2 / [R_x(e^{j\omega})R_y(e^{j\omega})]$
Interrelations	
$\gamma_x(l) = r_x(l) - \|\mu_x\|^2$	
$\gamma_{xy}(l) = r_{xy}(l) - \mu_x\mu_y^*$	
Properties	
Autocorrelation	**Auto-PSD**
$r_x(l)$ is nonnegative definite	$R_x(e^{j\omega}) \ge 0$ and real
$r_x(l) = r_x^*(-l)$	$R_x(e^{j\omega}) = R_x(e^{-j\omega})$ [real $x(n)$]
$\|r_x(l)\| \le r_x(0)$	$R_x(z) = R_x^*(1/z^*)$
$\|\rho_x(l)\| \le 1$	$R_x(z) = R_x(z^{-1})$ [real $x(n)$]
Cross-correlation	**Cross-PSD**
$r_{xy}(l) = r_{yx}^*(-l)$	
$\|r_{xy}(l)\| \le [r_x(0)r_y(0)]^{1/2} \le$	$R_{xy}(z) = R_{yx}^*(1/z^*)$
$[r_x(0)+r_y(0)]/2$	$0 \le \|G_{xy}(e^{j\omega})\| \le 1$
$\|\rho_{xy}(l)\| \le 1$	

2.2 Linear Systems with Stationary Random Inputs

This section deals with the processing of stationary random sequences using linear, time-invariant (LTI) systems. We focus on expressing the second-order statistical properties of the output in terms of the corresponding properties of the input and the characteristics of the system.

2.2.1 *Time-Domain Analysis*

The first question to ask when we apply a random signal to a system is, just what is the meaning of such an operation? We ask this because a random process is not just a single sequence but an ensemble of sequences (see Section 2.1). However, since each realization of the stochastic process is a deterministic signal, it is an acceptable input producing an output that is clearly a single realization of the output stochastic process. For an LTI system, each pair of input-output realizations is described by the convolution summation

$$y(n,\zeta)=\sum_{k=-\infty}^{\infty}h(k)x(n-k,\zeta) \tag{2.2.1}$$

If the sum in the right side of (2.2.1) exists for all ζ such that $\Pr\{\zeta\}=1$, then we say that we have almost-everywhere convergence or convergence with probability 1 (Papoulis 1991). The existence of such convergence is ruled by the following theorem (Brockwell and Davis 1991).

THEOREM 2.1. If the process $x(n,\zeta)$ is stationary with $E\{|x(n,\zeta)|\}<\infty$ and if the system is BIBO-stable, that is, $\sum_{-\infty}^{\infty}|h(k)|<\infty$, then the output $y(n,\zeta)$ of the system in (2.2.1) converges absolutely with probability 1, or

$$y(n,\zeta)=\sum_{k=-\infty}^{\infty}h(k)x(n-k,\zeta) \qquad \text{for all } \zeta\in A,\ \Pr\{A\}=1 \tag{2.2.2}$$

and is stationary. Furthermore, if $E\{|x(n,\zeta)|^2\}<\infty$, then $E\{|y(n,\zeta)|^2\}<\infty$ and $y(n,\zeta)$ converges in the mean square to the same limit and is stationary.

A less restrictive condition of finite power on the system impulse response $h(n)$ also guarantees the mean square existence of the output process, as stated in the following theorem.

THEOREM 2.2. If the process $x(n,\zeta)$ is zero-mean and stationary with $\sum_{l=-\infty}^{\infty}|r_x(l)|<\infty$, and if the system (2.2.1) satisfies the condition

$$\sum_{n=-\infty}^{\infty}|h(n)|^2=\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(e^{j\omega})|^2\,d\omega<\infty \tag{2.2.3}$$

then the output $y(n,\zeta)$ converges in the mean square sense and is stationary.

The above two theorems are applicable when input processes have finite variances. However, IID sequences with α-stable distributions have infinite variances. If the impulse response of the system in (2.2.1) decays fast enough, then the following theorem (Brockwell and Davis 1991) guarantees the absolute convergence of $y(n,\zeta)$ with probability 1. These issues are of particular importance for inputs with high variability and are discussed in Section 2.1.5.

THEOREM 2.3. Let $x(n,\zeta)$ be an IID sequence of random variables with α-stable distribution, $0<\alpha<2$. If the impulse response $h(n)$ satisfies

$$\sum_{n=-\infty}^{\infty}|h(n)|^{\delta}<\infty \qquad \text{for some } \delta\in(0,\alpha)$$

then the output $y(n,\zeta)$ in (2.2.1) converges absolutely with probability 1.

Clearly, a complete description of the output stochastic process $y(n)$ requires the computation of an infinite number of convolutions. Thus, a better alternative would be to determine the statistical properties of $y(n)$ in terms of the statistical properties of the input and the characteristics of the system. For Gaussian signals, which are used very often in practice, first- and second-order statistics are sufficient.

Output mean value. If $x(n)$ is stationary, its first-order statistic is determined by its mean value μ_x. To determine the mean value of the output, we take the expected value of both sides of (2.2.1):

$$\mu_y = \sum_{k=-\infty}^{\infty} h(k)E\{x(n-k)\} = \mu_x \sum_{k=-\infty}^{\infty} h(k) = \mu_x H(e^{j0}) \tag{2.2.4}$$

Since μ_x and $H(e^{j0})$ are constant, μ_y is also constant. Note that $H(e^{j0})$ is the dc gain of the spectrum.

Input-output cross-correlation. If we take complex conjugate of (2.2.1), premultiply it by $x(n+l)$, and take the expectation of both sides, we have

$$E\{x(n+l)y^*(n)\} = \sum_{k=-\infty}^{\infty} h^*(k)E\{x(n+l)x^*(n-k)\}$$

or

$$r_{xy}(l) = \sum_{k=-\infty}^{\infty} h^*(k)r_{xx}(l+k) = \sum_{m=-\infty}^{\infty} h^*(-m)r_{xx}(l-m)$$

Hence,

$$r_{xy}(l) = h^*(-l) * r_{xx}(l) \tag{2.2.5}$$

Similarly,

$$r_{yx}(l) = h(l) * r_{xx}(l) \tag{2.2.6}$$

Output autocorrelation. Postmultiplying both sides of (2.2.1) by $y^*(n-l)$ and taking the expectation, we obtain

$$E\{y(n)y^*(n-l)\} = \sum_{k=-\infty}^{\infty} h(k)E\{x(n-k)y^*(n-l)\} \tag{2.2.7}$$

or

$$r_{yy}(l) = \sum_{k=-\infty}^{\infty} h(k)r_{xy}(l-k) = h(l) * r_{xy}(l) \tag{2.2.8}$$

From (2.2.5) and (2.2.8) we get

$$r_y(l) = h(l) * h^*(-l) * r_x(l) \tag{2.2.9}$$

or

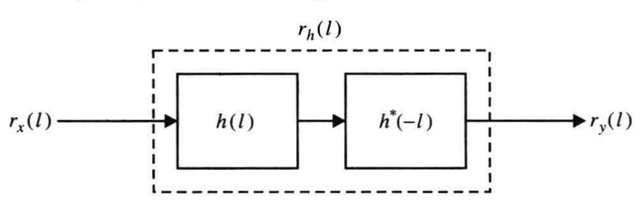
$$r_y(l) = r_h(l) * r_x(l) \tag{2.2.10}$$

where

$$r_h(l) \triangleq h(l) * h^*(-l) = \sum_{n=-\infty}^{\infty} h(n)h^*(n-l) \tag{2.2.11}$$

is the autocorrelation of the impulse response and is called the system correlation sequence.

Since μ_y is constant and $r_y(l)$ depends only on the lag l, the response of a stable system to a stationary input is also a stationary process. A careful examination of (2.2.10) shows that *when a signal $x(n)$ is filtered by an LTI system with impulse response $h(n)$ its autocorrelation is "filtered" by a system with impulse response equal to the autocorrelation of its impulse response*, as shown in Figure 2.4



FIRUGRE 2.4
An equivalent LTI system for autocorrelation filtration.

Output power. The power $E\{|y(n)|^2\}$ of the output process $y(n)$ is equal to $r_y(0)$, which from (2.2.9) and (2.2.10) and the symmetry property of $r_x(l)$ is

$$P_y = r_y(0) = r_h(l) * r_x(l)\big|_{l=0} = \sum_{k=-\infty}^{\infty} r_h(k) r_x(-k) = \sum_{k=-\infty}^{\infty} [h(k) * h^*(-k)] r_x(k)$$

$$= \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h(m) h^*(m-k) r_x(k) \tag{2.2.12}$$

$$= \sum_{k=-\infty}^{\infty} r_h(k) r_x(k) \tag{2.2.13}$$

or for FIR filters with $\boldsymbol{h} = [h(0) \ \ h(1) \ \ \cdots \ \ h(M-1)]^{\mathrm{T}}$, (2.2.12) can be written as

$$P_y = \boldsymbol{h}^{\mathrm{H}} \boldsymbol{R}_x \boldsymbol{h} \tag{2.2.14}$$

Finally, we note that when $\mu_x = 0$, we have $\mu_y = 0$ and $\sigma_y^2 = P_y$.

Output probability density function. Finding the probability density of the output of an LTI system is very difficult, except in some special cases. Thus, if $x(n)$ is a Gaussian process, then the output is also a Gaussian process with mean and autocorrelation given by (2.2.4) and (2.2.10). Also if $x(n)$ is IID, the probability density of the output is obtained by noting that $y(n)$ is a weighted sum of independent random variables. Indeed, the probability density of the sum of independent random variables is the convolution of their probability densities or the products of their characteristic functions. Thus if the input process is an IID stable process then the output process is also stable whose probability density can be computed by using characteristic functions.

2.2.2 *Frequency-Domain Analysis*

To obtain the output autoPSD and complex autoPSD, we recall that if $H(z) = Z\{h(n)\}$, then, for real $h(n)$,

$$Z\{h^*(-n)\} = H^*\left(\frac{1}{z^*}\right) \tag{2.2.15}$$

From (2.2.5), (2.2.6), and (2.2.7) we obtain

$$R_{xy}(z) = H^*\left(\frac{1}{z^*}\right) R_x(z) \tag{2.2.16}$$

$$R_{yx}(z) = H(z) R_x(z) \tag{2.2.17}$$

and

$$R_y(z) = H(z) H^*\left(\frac{1}{z^*}\right) R_x(z) \tag{2.2.18}$$

For a stable system, the unit circle $z = e^{j\omega}$ lies within the ROCs of $H(z)$ and $H(z^{-1})$. Thus,

$$R_{xy}(\mathrm{e}^{\mathrm{j}\omega}) = H^*(\mathrm{e}^{\mathrm{j}\omega}) R_x(\mathrm{e}^{\mathrm{j}\omega}) \tag{2.2.19}$$

$$R_{yx}(\mathrm{e}^{\mathrm{j}\omega}) = H(\mathrm{e}^{\mathrm{j}\omega}) R_x(\mathrm{e}^{\mathrm{j}\omega}) \tag{2.2.20}$$

and

$$R_y(\mathrm{e}^{\mathrm{j}\omega}) = H(\mathrm{e}^{\mathrm{j}\omega}) H^*(\mathrm{e}^{\mathrm{j}\omega}) R_x(\mathrm{e}^{\mathrm{j}\omega}) \tag{2.2.21}$$

or

$$R_y(\mathrm{e}^{\mathrm{j}\omega}) = |H(\mathrm{e}^{\mathrm{j}\omega})|^2 R_x(\mathrm{e}^{\mathrm{j}\omega}) \tag{2.2.22}$$

Thus, if we know the input and output autocorrelations or autospectral densities, we can determine the magnitude response of a system, but not its phase response. Only cross-correlation or cross-spectral densities can provide phase information [see (2.2.19) and (2.2.20)].

It can easily be shown that the power of the output is

$$E\{|y(n)|^2\} = r_{yy}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(\mathrm{e}^{\mathrm{j}\omega})|^2 R_x(\mathrm{e}^{\mathrm{j}\omega}) \mathrm{d}\omega \tag{2.2.23}$$

$$= \sum_{l=-\infty}^{l} r_x(l) r_h(l) \tag{2.2.24}$$

which is equivalent to (2.2.13).

Consider now a narrowband filter with frequency response

$$H(e^{j\omega}) = \begin{cases} 1 & \omega_c - \dfrac{\Delta\omega}{2} \le \omega \le \omega_c + \dfrac{\Delta\omega}{2} \\ 0 & \text{elsewhere} \end{cases} \tag{2.2.25}$$

The power of the filter output is

$$E\{|y(n)|^2\} = \frac{1}{2\pi} \int_{\omega_c - \Delta\omega/2}^{\omega_c + \Delta\omega/2} R_x(e^{j\omega}) d\omega \simeq R_x(e^{j\omega_c}) \tag{2.2.26}$$

assuming that $\Delta\omega$ is sufficiently small and that $R_x(e^{j\omega})$ is continuous at $\omega = \omega_c$. Since $E\{|y(n)|^2\} \ge 0$, $R_x(e^{j\omega_c})$ is also nonnegative for all ω_c and $\Delta\omega$, hence

$$R_x(e^{j\omega}) \ge 0 \qquad -\pi \le \omega \le \pi \tag{2.2.27}$$

Hence, the PSD $R_x(e^{j\omega})$ is nonnegative definite for any random sequence $x(n)$ real (or complex). Furthermore, $R_x(e^{j\omega}) d\omega/(2\pi)$, has the interpretation of power, or $R_x(e^{j\omega})$ is a power density as a function of frequency (in cycles per second). Table 2.2 shows various input-output relationships in both the time and frequency domains.

Table 2.2.
Second-order moments of stationary random sequences processed by linear,time-invariant systems.

Time domain	Frequency domain	z Domain
$y(n) = h(n) * x(n)$	Not available	Not available
$r_{yx}(l) = h(l) * r_x(l)$	$R_{yx}(e^{j\omega}) = H(e^{j\omega}) R_x(e^{j\omega})$	$R_{yx}(z) = H(z) R_x(z)$
$r_{xy}(l) = h^*(-l) * r_x(l)$	$R_{xy}(e^{j\omega}) = H^*(e^{j\omega}) R_x(e^{j\omega})$	$R_{xy}(z) = H^*(1/z^*) R_x(z)$
$r_y(l) = h(l) * r_{xy}(l)$	$R_y(e^{j\omega}) = H(e^{j\omega}) R_{xy}(e^{j\omega})$	$R_y(z) = H(z) R_{xy}(z)$
$r_y(l) = h(l) * h^*(-l) * r_x(l)$	$R_y(e^{j\omega}) = \|H(e^{j\omega})\|^2 R_x(e^{j\omega})$	$R_y(z) = H(z) H^*(1/z^*) R_x(z)$

2.2.3 Random Signal Memory

Given the "zero-memory" process $\omega(n) \sim IID(0, \sigma_\omega^2)$, we can introduce dependence by passing it through an LTI system. The extent and degree of the imposed dependence are dictated by the shape of the system's impulse response. The probability density of $\omega(n)$ is not explicitly involved. Suppose now that we are given the resulting linear process $x(n)$, and we want to quantify its memory. For processes with finite variance we can use the *correlation length*

$$L_c = \frac{1}{r_x(0)} \sum_{l=0}^{\infty} r_x(l) = \sum_{l=0}^{\infty} \rho_x(l)$$

which equals the area under the normalized autocorrelation sequence curve and shows the maximum distance at which two samples are significantly correlated.

An IID process has no memory and is completely described by its first-order density. A linear process has memory introduced by the impulse response of the generating system. If $\omega(n)$ has finite variance, the memory of the process is determined by the autocorrelation of the impulse response because $r_x(l) = \sigma_\omega^2 r_h(l)$. Also, the higher-order densities of the process are nonzero. Thus, the variability of the output—that is, what amplitudes takes the signal, how often, and how fast the amplitude changes from sample to sample—is the combined effect of the input probability density and the system memory.

DEFINITON 2.7. A stationary process $x(n)$ with finite variance is said to have *long memory* if there exist constants $\alpha, 0 < \alpha < 1$, and $C_r > 0$ such that

$$\lim_{l \to \infty} \frac{1}{C_r \sigma_x^2} r_x(l) l^\alpha = 1$$

This implies that the autocorrelation has fat or heavy tails, that is, asymptotically decays as a power law

$$\rho_x(l) \simeq C_r |l|^{-\alpha} \qquad \text{as } l \to \infty$$

and slowly enough that

$$\sum_{l=-\infty}^{\infty} \rho_x(l) = \infty$$

that is, a long-memory process has infinite correlation length. If

$$\sum_{l=-\infty}^{\infty} \rho_x(l) < \infty$$

we say that that the process has *short memory*. This is the case for autocorrelations that decay exponentially, for example, $\rho_x(l) = a^{|l|}, -1 < a < 1$.

An equivalent definition of long memory can be formulated in terms of the power spectrum (Beran 1994; Samorodnitsky and Taqqu 1994).

DEFINTION 2.8. A stationary process $x(n)$ with finite variance is said to have *long memory* if there exist constants $\beta, 0 < \beta < 1$, and $C_R > 0$ such that

$$\lim_{\omega \to 0} \frac{1}{C_R \sigma_x^2} R_x(e^{j\omega}) |\omega|^{\beta} = 1$$

This asymptotic definition implies that

$$R_x(e^{j\omega}) \simeq \frac{C_R \sigma_x^2}{|\omega|^{\beta}} \qquad \text{as } \omega \to 0$$

and

$$R_x(0) = \sum_{l=-\infty}^{\infty} r_x(l) = \infty$$

The first-order density determines the mean value and the variance of a process, whereas the second-order density determines the autocorrelation and power spectrum. There is a coupling between the probability density and the autocorrelation or power spectrum of a process. However, this coupling is not extremely strong because there are processes that have different densities and the same autocorrelation. Thus, we can have random signal models with short or long memory and low or high variability. Random signal models are discussed in Chapters 3.

2.2.4 General Correlation Matrices

We first begin with the properties of general correlation matrices. Similar properties apply to covariance matrices.

PROPERTY 2.2.1. The correlation matrix of a random vector $\mathbf{x}$ is conjugate symmetric or Hermitian, that is,

$$\boldsymbol{R}_x = \boldsymbol{R}_x^{\mathrm{H}} \tag{2.2.28}$$

Proof This follows easlily from (2.2.19).

PROPERTY 2.2.2. The correlation matrix of a random vector $\mathbf{x}$ is nonnegative definite (n.n.d.); or for every nonzero complex vector $\boldsymbol{w} = [w_1 \ w_2 \ \cdots \ w_M]^{\mathrm{T}}$, the quadratic form $\boldsymbol{w}^{\mathrm{H}} \boldsymbol{R}_x \boldsymbol{w}$ is nonnegative, that is,

$$\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} \geq 0 \tag{2.2.29}$$

Proof. To prove (2.2.29), we define the dot product

$$\alpha = \boldsymbol{w}^{\mathrm{H}} \boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}^* = \sum_{k=1}^{M} w_k^* x_k \tag{2.2.30}$$

The mean square value of the random variable α is

$$E\{|\alpha|^2\} = E\{\boldsymbol{w}^{\mathrm{H}} \boldsymbol{x}\boldsymbol{x}^{\mathrm{H}} \boldsymbol{w}\} = \boldsymbol{w}^{\mathrm{H}} E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\} \boldsymbol{w} = \boldsymbol{w}^{\mathrm{H}} \boldsymbol{R}_x \boldsymbol{w} \tag{2.2.31}$$

Since $E\{|\alpha|^2\} \geq 0$, if follows that $\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} \geq 0$. We also note that a matrix is called *positive definite* (p.d.) if $\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} > 0$.

Eigenvalues and eigenvectors of *R*

For a Hermitian matrix $\boldsymbol{R}$ we wish to find an $M \times 1$ vector $\mathbf{q}$ that satisfies the condition

$$\boldsymbol{R}\boldsymbol{q} = \lambda \boldsymbol{q} \tag{2.2.32}$$

where λ is a constant. This condition implies that the linear transformation performed by matrix $\boldsymbol{R}$ does not

change the direction of vector q. Thus Rq is a *direction-invariant* mapping. To determine the vector q, we write (2.2.32) as

$$(R-\lambda I)q = 0 \tag{2.2.33}$$

where I is the $M\times M$ identity matrix and 0 is an $M\times 1$ vector of zeros. Since q is arbitrary, the only way (2.2.33) is satisfied is if the determinant of $R-\lambda I$ equals zero, that is,

$$\det(R-\lambda I) = 0 \tag{2.2.34}$$

This equation is an M th-order polynomial in λ and is called the *characteristic equation* of R. It has M roots $\{\lambda_i\}_{i=1}^{M}$, called *eigenvalues*, which, in general, are distinct. If (2.2.34) has repeated roots, then R is said to have *degenerate* eigenvalues. For each eigenvalue λ_i we can satisfy (2.2.32)

$$Rq_i = \lambda_i q_i \qquad i = 1,\cdots,M \tag{2.2.35}$$

where the q_i are called *eigenvectors* of R. Therefore, the $M\times M$ matrix R has M eigenvectors. To uniquely determine $\mathbf{q}_i$, we use (2.2.35) along with the normality condition that $\|q_i\| = 1$. A MATLAB function `[Lambda,Q] = eig(R)` is available to compute eigenvalues and eigenvectors of R.

There are further properties of the autocorrelation matrix R based on its eigenanalysis, which we describe below. Consider a matrix R that is Hermitian and nonnegative definite $(w^H R w \ge 0)$ with eigenvalues $\{\lambda_i\}_{i=1}^{M}$ and eigenvectors $\{q_i\}_{i=1}^{M}$.

PROPERTY 2.2.3. The matrix $R^k (k=1,2,\cdots)$ has eigenvalues $\lambda_1^k, \lambda_2^k, \ldots, \lambda_M^k$.

Proof. See Problem 2.16.

PROPERTY 2.2.4. If the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ are distinct, the corresponding eigenvectors $\{q_i\}_{i=1}^{M}$ are *linearly independent.*

Proof. This property can be proved by using Property 2.2.3 Given M not-all-zero scalars $\{\alpha_i\}_{i=1}^{M}$, if

$$\sum_{i=1}^{M} \alpha_i q_i = 0 \tag{2.2.36}$$

then the eigenvectors $\{q_i\}_{i=1}^{M}$ are said to be *linearly dependent*. Assume that (2.2.36) is true for some not-all-zero scalars $\{\alpha_i\}_{i=1}^{M}$ and that the eigenvalues $\{\lambda_i\}_{i=1}^{M}$ are distinct. Now multiply (2.2.36) repeatedly by R^k, $k=0,\cdots,M-1$ and use Property 2.2.3 to obtain

$$\sum_{i=1}^{M} \alpha_i R^k q_i = \sum_{i=1}^{M} \alpha_i \lambda_i^k q_i = 0 \qquad k = 0,\cdots,M-1 \tag{2.2.37}$$

which can be arranged in a matrix format for $i = 1,\cdots,M$ as

$$\begin{bmatrix}\alpha_1 q_1 & \alpha_2 q_2 & \alpha_3 q_3 & \cdots & \alpha_M q_M\end{bmatrix}\begin{bmatrix}1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{M-1}\\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^{M-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \lambda_M & \lambda_M^2 & \cdots & \lambda_M^{M-1}\end{bmatrix} = 0 \tag{2.2.38}$$

Since all the λ_i are distinct, the matrix containing the λ_i in (2.2.38) above is nonsingular. This matrix is called a *Vandermonde* matrix. Therefore, premultiplying both sides of (2.2.38) by the inverse of the Vandermonde matrix, we obtain

$$[\alpha_1 q_1 \;\; \alpha_2 q_2 \;\; \alpha_3 q_3 \;\; \cdots \;\; \alpha_M q_M] = 0 \tag{2.2.39}$$

Since eigenvectors $\{q_i\}_{i=1}^{M}$ are not zero vectors, the only way (2.2.39) can be satisfied is if all $\{\alpha_i\}_{i=1}^{M}$ are zero. This implies that (2.2.36) cannot be satisfied for any set of not-all-zero scalars $\{\alpha_i\}_{i=1}^{M}$, which further implies that $\{q_i\}_{i=1}^{M}$ are linearly independent.

PROPERTY 2.2.5. The eigenvalues $\{\lambda_i\}_{i=1}^{M}$ are real and *nonnegative*.

Proof. From (2.2.35), we have

$$q_i^{\mathrm{H}} R q_i = \lambda_i q_i^{\mathrm{H}} q_i \qquad i = 1,2,\cdots,M \tag{2.2.40}$$

Since R is positive semidefinite, the quadratic form $q_i^H R q_i \ge 0$. Also since $q_i^{\mathrm{H}} q_i$ is an inner product, $q_i^{\mathrm{H}} q_i > 0$. Hence

$$\lambda_i = \frac{q_i^{\mathrm{H}} R q_i}{q_i^H q_i} \ge 0 \qquad i = 1,2,\cdots,M \tag{2.2.41}$$

Furthermore, if R is positive definite, then $\lambda_i > 0$ for all $1 \le i \le M$. The quotient in (2.2.41) is a useful quantity and is known as the *Raleigh quotient* of vector q_i.

PROPERTY 2.2.6. If the eigenvalues $\{\lambda_i\}_{i=1}^{M}$ are distinct, then the corresponding eigenvectors are orthogonal to one another, that is,

$$\lambda_i \neq \lambda_j \Rightarrow \boldsymbol{q}_i^{\mathrm{H}} \boldsymbol{q}_j = 0 \qquad \text{for } i \neq j \tag{2.2.42}$$

Proof. Consider (2.2.35). We have

$$\boldsymbol{R}\boldsymbol{q}_i = \lambda_i \boldsymbol{q}_i \tag{2.2.43}$$

and

$$\boldsymbol{R}\boldsymbol{q}_j = \lambda_j \boldsymbol{q}_j \tag{2.2.44}$$

For some $i \neq j$, premultiplying both sides of (2.2.43) by $\boldsymbol{q}_j^{\mathrm{H}}$, we obtain

$$\boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{R}\boldsymbol{q}_i = \boldsymbol{q}_j^{\mathrm{H}} \lambda_i \boldsymbol{q}_i = \lambda_i \boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{q}_i \tag{2.2.45}$$

Taking the conjugate transpose of (2.2.44), using the Hermitian property (2.2.35) of $\boldsymbol{R}$, and using the realness Property 2.2.5 of eigenvalues, we get

$$\boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{R} = \lambda_j \boldsymbol{q}_j^{\mathrm{H}} \tag{2.2.46}$$

Now postmultiplying (2.2.46) by q_i and comparing with (2.2.45), we conclude that

$$\lambda_i \boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{q}_i = \lambda_j \boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{q}_i \qquad \text{or} \qquad (\lambda_i - \lambda_j) \boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{q}_i = 0 \tag{2.2.47}$$

Since the eigenvalues are assumed to be distinct, the only way (2.2.47) can be satisfied is if $\boldsymbol{q}_j^{\mathrm{H}} \boldsymbol{q}_i = 0$ for $i \neq j$, which further proves that the corresponding eigenvectors are orthogonal to one another.

PROPERTY 2.2.7. Let $\{\boldsymbol{q}_i\}_{i=1}^{M}$ be an orthonormal set of eigenvectors corresponding to the distinct eigenvalues $\{\lambda_i\}_{i=1}^{M}$ of an $M \times M$ correlation matrix $\boldsymbol{R}$. Then $\boldsymbol{R}$ can be diagonalized as follows:

$$\Lambda = \boldsymbol{Q}^{\mathrm{H}} \boldsymbol{R} \boldsymbol{Q} \tag{2.2.48}$$

where the orthonormal matrix $\boldsymbol{Q} \triangleq [\boldsymbol{q}_1 \cdots \boldsymbol{q}_M]$ is known as an *eigenmatrix* and Λ is an $M \times M$ diagonal eigenvalue matrix, that is,

$$\Lambda \triangleq diag(\lambda_1, \cdots, \lambda_M) \tag{2.2.49}$$

Proof. Arranging the vectors in (2.2.35) in a matrix format, we obtain

$$[\boldsymbol{R}\boldsymbol{q}_1 \;\; \boldsymbol{R}\boldsymbol{q}_2 \;\; \cdots \;\; \boldsymbol{R}\boldsymbol{q}_M] = [\lambda_1 \boldsymbol{q}_1 \;\; \lambda_2 \boldsymbol{q}_2 \;\; \cdots \;\; \lambda_M \boldsymbol{q}_M]$$

which, by using the definitions of $\boldsymbol{Q}$ and Λ, can be further expressed as

$$\boldsymbol{R}\boldsymbol{Q} = \boldsymbol{Q}\Lambda \tag{2.2.50}$$

Since $\boldsymbol{q}_i, i = 1, \cdots, M$, is an orthonormal set of vectors, the eigenmatrix $\boldsymbol{Q}$ is unitary, that is, $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{\mathrm{H}}$. Now premultiplying both sides of (2.2.50) by $\boldsymbol{Q}^{\mathrm{H}}$, we obtain the desired result.

This diagonalization of the autocorrelation matrix plays an important role in filtering and estimation theory, as we shall see later. From (2.2.48) the correlation matrix $\boldsymbol{R}$ can also be written as

$$\boldsymbol{R} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^{\mathrm{H}} = \lambda_1 \boldsymbol{q}_1 \boldsymbol{q}_1^{\mathrm{H}} + \cdots + \lambda_M \boldsymbol{q}_M \boldsymbol{q}_M^{\mathrm{H}} = \sum_{m=1}^{M} \lambda_m \boldsymbol{q}_m \boldsymbol{q}_m^{\mathrm{H}} \tag{2.2.51}$$

which is known as the *spectral theorem*, or *Mercer's theorem*. If $\boldsymbol{R}$ is positive definite (and hence invertible), its inverse is given by

$$\boldsymbol{R}^{-1} = (\boldsymbol{Q}\Lambda\boldsymbol{Q}^{\mathrm{H}})^{-1} = \boldsymbol{Q}\Lambda^{-1}\boldsymbol{Q}^{\mathrm{H}} = \sum_{m=1}^{M} \frac{1}{\lambda_m} \boldsymbol{q}_m \boldsymbol{q}_m^{\mathrm{H}} \tag{2.2.52}$$

because Λ is a diagonal matrix.

PROPERTY 2.2.8. The trace of $\boldsymbol{R}$ is the summation of all eigenvalues, that is,

$$tr(\boldsymbol{R}) = \sum_{i=1}^{M} \lambda_i \tag{2.2.53}$$

Proof. See Problem 2.17.

PROPERTY 2.2.9. The determinant of $\boldsymbol{R}$ is equal to the product of all eigenvalues, that is,

$$\det \boldsymbol{R} = |\boldsymbol{R}| = \prod_{i=1}^{M} \lambda_i = |\Lambda| \tag{2.2.54}$$

Proof. See Problem 2.18.

PROPERTY 2.2.10. Determinants of $\boldsymbol{R}$ and Γ are related by

$$|\boldsymbol{R}|=|\Gamma|(1+\mu_x^{\mathrm{H}}\Gamma_{\mathbf{x}}\mu_x) \tag{2.2.55}$$

Proof. See Problem 2.19.

$\boldsymbol{\Gamma}_x$ is the autocovariance matrix, defined by

$$\Gamma_x = E\{[X(\xi)-\mu_X][X(\xi)-\mu_X]^{\mathrm{H}}\} = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1M} \\ \vdots & \ddots & \vdots \\ \gamma_{M1} & \cdots & \gamma_{MM} \end{bmatrix}$$

2.2.5 *Correlation Matrices from Random Processes*

A stochastic process can also be represented as a random vector, and its second-order statistics given by the mean vector and the correlation matrix. Obviously, these quantities are functions of the index n. Let an $M\times1$ random vector $\mathbf{x}(n)$ be derived from the random process $x(n)$ as follows:

$$\boldsymbol{x}(n) \triangleq [x(n)\ x(n-1)\ \cdots\ x(n-M+1)]^{\mathrm{T}} \tag{2.2.56}$$

Then its mean is given by an $M\times1$ vector

$$\mu_x(n)=[\mu_x(n)\ \mu_x(n-1)\ \cdots\ \mu_x(n-M+1)]^{\mathrm{T}} \tag{2.2.57}$$

and the correlation by an $M\times M$ matrix

$$\boldsymbol{R}_x(n)=\begin{bmatrix} r_x(n,n) & \cdots & r_x(n,n-M+1) \\ \vdots & \ddots & \vdots \\ r_x(n-M+1,n) & \cdots & r_x(n-M+1,n-M+1) \end{bmatrix} \tag{2.2.58}$$

Clearly, $\boldsymbol{R}_x(n)$ is Hermitian since $r_x(n-i,n-j)=r_x^*(n-j,n-i), 0\le i,j\le M-1$. This vector representation will be useful when we discuss optimum filters.

Correlation matrices of stationary processes

The correlation matrix $\boldsymbol{R}_x(n)$ of a general stochastic process $x(n)$ is a Hermitian $M\times M$ matrix defined in (2.2.58) with elements $r_x(n-i,n-j)=E\{x(n-i)x^*(n-j)\}$. For stationary processes this matrix has an interesting additional structure. First, $\boldsymbol{R}_x(n)$ is a constant matrix $\boldsymbol{R}_x$; then using (2.1.24), we have

$$r_x(n-i,n-j)=r_x(j-i)=r_x(l) \quad (l\triangleq j-i) \tag{2.2.59}$$

Finally, by using conjugate symmetry $r_x(l)=r_x^*(-l)$, the matrix $\mathbf{R}_x$ is given by

$$\boldsymbol{R}_x=\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(M-1) \\ r_x^*(1) & r_x(0) & r_x(1) & \cdots & r_x(M-2) \\ r_x^*(2) & r_x^*(1) & r_x(0) & \cdots & r_x(M-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x^*(M-1) & r_x^*(M-2) & r_x^*(M-3) & \cdots & r_x(0) \end{bmatrix} \tag{2.2.60}$$

It can be easily seen that $\boldsymbol{R}_x$ is Hermitian and Toeplitz.[3] Thus, the autocorrelation matrix of a stationary process is Hermitian, nonnegative definite, and Toeplitz.

Eigenvalue spread and spectral dynamic range

The ill conditioning of a matrix $\mathbf{R}_x$ increases with its condition number $\mathcal{X}(\boldsymbol{R}_x)=\lambda_{\max}/\lambda_{\min}$. When $\boldsymbol{R}_x$ is a

---

[3] A matrix is called *Toeplitz* if the elements along each diagonal, parallel to the main diagonal, are equal.

correlation matrix of a stationary process, then $\mathcal{X}(R_x)$ is bounded from above by the dynamic range of the PSD $R_x(e^{j\omega})$ of the process $x(n)$. The larger the spread in eigenvalues, the wider (or less flat) the variation of the PSD function. This is also related to the dynamic range or to the data spread in $x(n)$ and is a useful measure in practice. This result is given by the following theorem, in which we have dropped the subscript of $R_x(e^{j\omega})$ for clarity.

THEOREM 2.4. Consider a zero-mean stationary random process with autoPSD

$$R(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}$$

then

$$\min_{\omega} R(e^{j\omega}) \le \lambda_i \le \max_{\omega} R(e^{j\omega}) \qquad \text{for all } i = 1, 2, \cdots, M \tag{2.2.61}$$

Proof. From (2.2.41) we have

$$\lambda_i = \frac{\boldsymbol{q}_i^H \boldsymbol{R} \boldsymbol{q}_i}{\boldsymbol{q}_i^T \boldsymbol{q}_i} \tag{2.2.62}$$

Consider the quadratic form

$$\boldsymbol{q}_i^H \boldsymbol{R} \boldsymbol{q}_i = \sum_{k=1}^{M} \sum_{l=1}^{M} q_i(k) r(l-k) q_i(l)$$

where $\boldsymbol{q}_i = [q_i(1)\ q_i(2)\ \cdots\ q_i(M)]^T$. Using (2.1.41) and the stationarity of the process, we obtain

$$\begin{aligned} \boldsymbol{q}_i^H \boldsymbol{R} \boldsymbol{q}_i &= \frac{1}{2\pi} \sum_k \sum_l q_i^*(k) q_i(l) \int_{-\pi}^{\pi} R(e^{j\omega}) e^{j\omega(l-k)} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} R(e^{j\omega}) \left[\sum_{k=1}^{M} q_i^*(k) e^{-j\omega k} \right] \left[\sum_{l=1}^{M} q_i(l) e^{j\omega l} \right] d\omega \end{aligned} \tag{2.2.63}$$

or

$$\boldsymbol{q}_i^H \boldsymbol{R} \boldsymbol{q}_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} R(e^{j\omega}) |Q(e^{j\omega})|^2 d\omega \tag{2.2.64}$$

Similarly, we have

$$\boldsymbol{q}_i^T \boldsymbol{q}_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega \tag{2.2.65}$$

Substituting (2.2.64) and (2.2.65) in (2.2.62), we obtain

$$\lambda_i = \frac{\int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 R(e^{j\omega}) d\omega}{\int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega} \tag{2.2.66}$$

However, since $R(e^{j\omega}) \ge 0$, we have the following inequality:

$$\min_{\omega} R(e^{j\omega}) \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega \le \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 R(e^{j\omega}) d\omega \le \max_{\omega} R(e^{j\omega}) \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega$$

from which we easily obtain the desired result. The above result also implies that

$$\mathcal{X}(\mathbf{R}) \triangleq \frac{\lambda_{\max}}{\lambda_{\min}} \le \frac{\max_{\omega} R(e^{j\omega})}{\min_{\omega} R(e^{j\omega})} \tag{2.2.67}$$

which becomes equality as $M \to \infty$.

2.3 Innovations Representation of Random Vectors

In many practical and theoretical applications, it is desirable to represent a random vector (or sequence) with a linearly equivalent vector (or sequence) consisting of uncorrelated components. If $\boldsymbol{x}$ is a correlated random vector and if $\boldsymbol{A}$ is a nonsingular matrix, then the linear transformation

$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{x} \tag{2.3.1}$$

results in a random vector $\boldsymbol{w}$ that contains the same "information" as $\boldsymbol{x}$, and hence random vectors $\boldsymbol{x}$ and $\boldsymbol{w}$ are said to be linearly equivalent. Furthermore, if $\mathbf{w}$ is an uncorrelated random vector, then each component w_i of

w can be thought of as *adding* "new" information (or *innovation*) to w that is not present in the remaining components. Such a representation is called an *innovations representation* and provides additional insight into the understanding of random vectors and sequences. Additionally, it can simplify many theoretical derivations and can result in computationally efficient implementations.

Since Γ_w must be a diagonal matrix, we need to diagonalize the Hermitian, positive definite matrix Γ_x through the transformation matrix A. There are two approaches to this diagonalization. One approach is to use the eigenanalysis presented in Section 2.2.4, which results in the well-known Karhunen-Loève (KL) transform. The other approach is to use triangularization methods from linear algebra, which leads to the LDU (UDL) and LU (UL) decompositions. These vector techniques can be further extended to random sequences that give us the KL expansion and the spectral factorizations, respectively.

Here, only the transformation using eigendecomposition are discassed.

Transformations Using Eigendecomposition

Let $\mathbf{x}$ be a random vector with mean vector $\mu_{\mathbf{x}}$ and covariance matrix $\Gamma_{\mathbf{x}}$. The linear transformation

$$x_0 = x - \mu_{\mathbf{x}} \tag{2.3.2}$$

results in a zero-mean vector x_0 with correlation (and covariance) matrix equal to Γ_x. This transformation shifts the origin of the M-dimensional coordinate system to the mean vector. We will now consider the zero-mean random vector x_0 for further transformations.

Orthonormal transformation

Let Q_x be the eigenmatrix of Γ_x, and let us choose Q_x^{H} as our linear transformation matrix A. Consider

$$w = Q_x^{\mathrm{H}} x_0 = Q_x^{\mathrm{H}}(x - \mu_x) \tag{2.3.3}$$

Then

$$\mu_w = Q_x^{\mathrm{H}}(E\{x_0\}) = 0 \tag{2.3.4}$$

and

$$\Gamma_w = R_w = E\{Q_x^{\mathrm{H}} x_0 x_0^{\mathrm{H}} Q_x\} = Q_x^{\mathrm{H}} \Gamma_x \mathbf{Q}_{\mathbf{x}} = \Lambda_x \tag{2.3.5}$$

Since Λ_x is diagonal, Γ_w is also diagonal, and hence this transformation has some interesting properties:

1. The random vector w has zero mean, and its components are mutually uncorrelated (and hence orthogonal). Furthermore, if x is $N(\mu_x, \Gamma_x)$, then $\mathbf{w}$ is $N(0, \Lambda_x)$ with independent components.
2. The variances of random variables $w_i, i = 1, \cdots, M$, are equal to the eigenvalues of Γ_x.
3. Since the transformation matrix $A = Q_x^{\mathrm{H}}$ is orthonormal, the transformation is called an ***orthonormal transformation*** and the distance measure

$$d^2(x_0) \triangleq x_0^{\mathrm{H}} \Gamma_x^{-1} x_0 \tag{2.3.6}$$

is preserved under the transformation. This distance measure is also known as the ***Mahalanobis distance***; and in the case of normal random vectors, it is related to the log-likelihood function.

4. Since $w = Q_x^{\mathrm{H}}(x - \mu_x)$, we have

$$w_i = q_i^{\mathrm{H}}(x - \mu_x) = \| x - \mu_x \| \cos[\measuredangle(x - \mu_x, q_i)] \qquad i = 1, \cdots, M \tag{2.3.7}$$

which is the projection of $x - \mu_x$ onto the unit vector q_i. Thus w represents x in a new coordinate system that is shifted to μ_x and spanned by $q_i, i = 1, \cdots, M$. A geometric interpretation of this transformation for a two-dimensional case is shown in Figure 2.5, which shows a contour of $d^2(x_0) = x^{\mathrm{H}} \Gamma_x^{-1} x = w^{\mathrm{H}} \Lambda_x^{-1} w$ in the x and w coordinate systems $(w = Q_x^{\mathrm{H}} x)$.

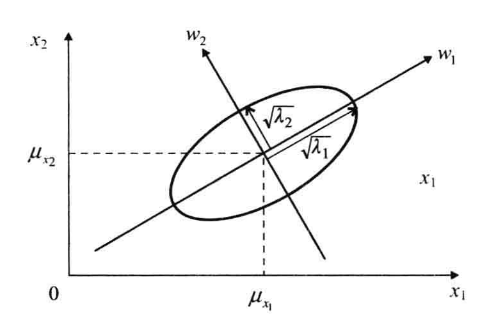


FIGURE 2.5
Orthogonal transformation in two dimensions.

Isotropic transformation

In the above orthonormal transformation, the autocorrelation matrix $\boldsymbol{R}_w$ is diagonal but not an identity matrix $\mathbf{I}$. This can be achieved by an additional linear mapping of $\Lambda_x^{-1/2}$. Let

$$\boldsymbol{y} = \Lambda_x^{-1/2}\boldsymbol{w} = \Lambda_x^{-1/2}\boldsymbol{Q}_x^{\mathrm{H}}\boldsymbol{x}_0 = \Lambda_x^{-1/2}\boldsymbol{Q}_x^{\mathrm{H}}(\boldsymbol{x}-\boldsymbol{\mu}_x) \tag{2.3.8}$$

Then

$$\boldsymbol{R}_y = \Lambda_x^{-1/2}\boldsymbol{Q}_x^{\mathrm{H}}\Gamma_x\boldsymbol{Q}_x\Lambda_x^{-1/2} = \Lambda_x^{-1/2}\Lambda_x\Lambda_x^{-1/2} = \boldsymbol{I} \tag{2.3.9}$$

This is called an *isotropic transformation* because *all* components of $\mathbf{y}$ are zero-mean, uncorrelated random variables with unit variance.[4] The geometric interpretation of this transformation for a two-dimensional case is shown in Figure 2.6. It clearly shows that there is not only a shift and rotation but also a scaling of the coordinate axis so that the distribution is equal in all directions, that is, it is direction-invariant. Because the transformation $\boldsymbol{A} = \Lambda_x^{-1/2}\boldsymbol{Q}_x^{\mathrm{H}}$ is orthogonal but not orthonormal, the distance measure $d^2(\boldsymbol{x}_0)$ is not preserved under this mapping. Since the correlation matrix after this transformation is an identity matrix $\boldsymbol{I}$, it is invariant under any orthonormal mapping, that is,

$$\boldsymbol{Q}^{\mathrm{H}}\boldsymbol{I}\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{H}}\boldsymbol{Q} = \boldsymbol{I} \tag{2.3.10}$$

This fact can be used for simultaneous diagonalization of two Hermitian matrices.

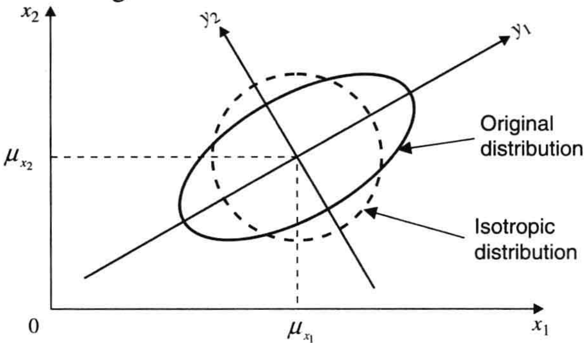


FIGURE 2.6
Isotropic transformation in two dimensions.

EXAMPLE 2.3.1. Consider a stationary sequence with correlation matrix

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$$

where $-1 < a < 1$. The eigenvalues

$$\lambda_1 = 1 + a \qquad \lambda_2 = 1 - a$$

are obtained from the characteristic equation

$$\det(\boldsymbol{R}_x - \lambda\boldsymbol{I}) = \det\begin{bmatrix} 1-\lambda & a \\ a & 1-\lambda \end{bmatrix} = (1-\lambda)^2 - a^2 = 0$$

To find the eigenvector $\boldsymbol{q}_1$, we solve the linear system

$$\begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}\begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \end{bmatrix} = (1+a)\begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \end{bmatrix}$$

which gives $q_1^{(1)} = q_2^{(1)}$. Similarly, we find that $q_1^{(2)} = -q_2^{(2)}$. If we normalize both vectors to unit length, we obtain the eigenvectors

[4] In the literature, an isotropic transformation is also known as a *whitening* transformation. We believe that this terminology is not accurate because both vectors $\boldsymbol{Q}_x^{\mathrm{H}}\boldsymbol{x}_0$ and $\Lambda_x^{-1/2}\boldsymbol{Q}_x^{\mathrm{H}}\boldsymbol{x}_0$ have uncorrelated coefficients.

$$q_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \qquad q_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

From the above results we see that $\det \mathbf{R}_x = 1 - a^2 = \lambda_1\lambda_2$ and $\mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{I}$, where $\mathbf{Q} = [\mathbf{q}_1\ \mathbf{q}_2]$.

2.4 Principles of Estimation Theory

The key assumption underlying our discussion up to this point was that the probability distributions associated with the problem under consideration were known. As a result, all required probabilities, autocorrelation sequences, and PSD functions either could be derived from a set of assumptions about the involved random processes or were given a priori. However, in most practical applications, this is the exception rather than the rule. Therefore, the properties and parameters of random variables and random processes should be obtained by collecting and analyzing finite sets of measurements. In this section, we introduce some basic concepts of estimation theory that will be used repeatedly in the rest of the book. Complete treatments of estimation theory can be found in Kay (1993), Helstrom (1995), Van Trees (1968), and Papoulis (1991).

2.4.1 Properties of Estimators

Suppose that we collect N observations $\{x(n)\}_0^{N-1}$ from a stationary stochastic process and use them to estimate a parameter θ (which we assume to be real-valued) of the process using some function $\hat{\theta}[\{x(n)\}_0^{N-1}]$. The same results can be used for a set of measurements $\{x_k(n)\}_1^N$ obtained from N sensors sampling stochastic processes with the same distributions. The function $\hat{\theta}[\{x(n)\}_0^{N-1}]$ is known as an *estimator* whereas the value taken by the estimator, using a particular set of observations, is called a *point estimate* or simply an *estimate*. The intention of the estimator design is that the estimate should be as close to the true value of the parameter as possible. However, if we use another set of observations or a different number of observations from the same set, it is highly unlikely that we will obtain the same estimate. As an example of an estimator, consider estimating the mean μ_x of a stationary process $x(n)$ from its N observations $\{x(n)\}_0^{N-1}$. Then the natural estimator is a simple arithmetic average of these observations, given by

$$\hat{\mu}_x = \hat{\theta}[\{x(n)\}_0^{N-1}] = \frac{1}{N}\sum_{n=0}^{N-1} x(n) \tag{2.4.1}$$

Similarly, a natural estimator of the variance σ_x^2 of the process $x(n)$ would be

$$\hat{\sigma}_x^2 = \hat{\theta}[\{x(n)\}_0^{N-1}] = \frac{1}{N}\sum_{n=0}^{N-1} [x(n) - \hat{\mu}_x]^2 \tag{2.4.2}$$

If we repeat this procedure a large number of times, we will obtain a large number of estimates, which can be used to generate a histogram showing the distribution of the estimates. Before the collection of observations, we would like to describe all sets of data that can be obtained by using the random variables $\{x(n,\zeta)\}_0^{N-1}$. The obtained set of N observations $\{x(n)\}_0^{N-1}$ can thus be regarded as one realization of the random variables $\{x(n)\}_0^{N-1}$ defined on an N-dimensional sample space. In this sense, the estimator $\hat{\theta}[\{x(n,\zeta)\}_0^{N-1}]$ becomes a random variable whose distribution can be obtained from the joint distribution of the random variables $\{x(n)\}_0^{N-1}$. This distribution is called the *sampling distribution* of the estimator and is a fundamental concept in estimation theory because it provides all the information we need to evaluate the quality of an estimator.

The sampling distribution of a "good" estimator should be concentrated as closely as possible about the parameter that it estimates. To determine how "good" an estimator is and how different estimators of the same parameter compare with one another, we need to determine their sampling distributions. Since it is ***not*** always possible to derive the exact sampling distributions, we have to resort to properties that use the lower-order moments (mean, variance, mean square error) of the estimator.

Bias of estimator. The *bias* of an estimator $\hat{\theta}$ of a parameter θ is defined as

$$B(\hat{\theta}) \triangleq E[\hat{\theta}] - \theta \tag{2.4.3}$$

while the *normalized* bias is defined as

$$\varepsilon_b \triangleq \frac{B(\hat{\theta})}{\theta} \qquad \theta \neq 0 \tag{2.4.4}$$

When $B(\hat{\theta})=0$, the estimator is said to be ***unbiased*** and the pdf of the estimator is centered exactly at the true value θ. Generally, one should select estimators that are unbiased such as the mean estimator in (2.4.1) or very nearly unbiased such as the variance estimator in (2.4.2). However, it is not always wise to select an unbiased estimator, as we will see below and in Section 4.2 on the estimation of autocorrelation sequences.

Variance of estimator. The *variance* of the estimator $\hat{\theta}$ is defined by

$$var(\hat{\theta})=\sigma_{\hat{\theta}}^2 \triangleq E\{|\hat{\theta}-E\{\hat{\theta}\}|^2\} \tag{2.4.5}$$

which measures the spread of the pdf of $\hat{\theta}$ around its average value. Therefore, one would select an estimator with the smallest variance. However, this selection is not always compatible with the small bias requirement. As we will see below, reducing variance may result in an increase in bias. Therefore, a balance between these two conflicting requirements is required, which is provided by the mean square error property. The *normalized standard deviation* (also called the coefficient of variation) is defined by

$$\varepsilon_r \triangleq \frac{\sigma_{\hat{\theta}}}{\theta} \qquad \theta \neq 0 \tag{2.4.6}$$

Mean square error. The *mean square error* (*MSE*) of the estimator is given by

$$MSE(\theta)=E\{|\hat{\theta}-\theta|^2\}=\sigma_{\hat{\theta}}^2+|B|^2 \tag{2.4.7}$$

Indeed, we have

$$MSE(\theta)=E\{|\theta-E\{\hat{\theta}\}-(\hat{\theta}-E\{\hat{\theta}\})|^2\}$$

$$=E\{|\theta-E\{\hat{\theta}\}|^2\}+E\{|\hat{\theta}-E\{\hat{\theta}\}|^2\} \tag{2.4.8}$$

$$-(\theta-E\{\hat{\theta}\})E\{(\hat{\theta}-E\{\hat{\theta}\})^*\}-(\theta-E\{\hat{\theta}\})^*E\{\hat{\theta}-E\{\hat{\theta}\}\}$$

$$=|\theta-E\{\hat{\theta}\}|^2+E\{|\hat{\theta}-E\{\hat{\theta}\}|^2\} \tag{2.4.9}$$

which leads to (2.4.7) by using (2.4.3) and (2.4.5). Ideally, we would like to minimize the MSE, but this minimum is not always zero. Hence minimizing variance can increase the bias. The *normalized MSE* is defined as

$$\varepsilon \triangleq \frac{MSE(\theta)}{\theta} \qquad \theta \neq 0 \tag{2.4.10}$$

Cramér-Rao lower bound. If it is possible to minimize the MSE when the bias is zero, then clearly the variance is also minimized. Such estimators are called *minimum variance unbiased* estimators, and they attain an important minimum bound on the variance of the estimator, called the *Cramér-Rao lower bound* (CRLB), or *minimum variance bound*. If $\hat{\theta}$ is unbiased, then it follows that $E\{\hat{\theta}-\theta\}=0$, which may be expressed as

$$\int_{-\infty}^{\infty}\cdots\int(\hat{\theta}-\theta)f_{\mathbf{x};\theta}(\mathbf{x};\theta)\,d\mathbf{x}=0 \tag{2.4.11}$$

where $\mathbf{x}(\zeta)=[x_1(\zeta),x_2(\zeta),\cdots,x_N(\zeta)]^T$ and $f_{\mathbf{x};\theta}(\mathbf{x};\theta)$ is the joint density of $\mathbf{x}(\zeta)$, which depends on a fixed but unknown parameter θ. If we differentiate (2.4.11) with respect to θ, assuming real-valued $\hat{\theta}$, we obtain

$$0=\int_{-\infty}^{\infty}\cdots\int\frac{\partial}{\partial\theta}[(\hat{\theta}-\theta)f_{\mathbf{x};\theta}(\mathbf{x};\theta)]\,d\mathbf{x}=\int_{-\infty}^{+\infty}\cdots\int(\hat{\theta}-\theta)\frac{\partial f_{\mathbf{x};\theta}(\mathbf{x};\theta)}{\partial\theta}\,d\mathbf{x}-1 \tag{2.4.12}$$

Using the fact

$$\frac{\partial\ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)]}{\partial\theta}=\frac{1}{f_{\mathbf{x};\theta}(\mathbf{x};\theta)}\frac{\partial f_{\mathbf{x};\theta}(\mathbf{x};\theta)}{\partial\theta}$$

or
$$\frac{\partial f_{x;\theta}(x;\theta)}{\partial \theta} = \frac{\partial \ln[f_{x;\theta}(x;\theta)]}{\partial \theta} f_{x;\theta}(x;\theta) \tag{2.4.13}$$

and substituting (2.4.13) in (2.4.12), we get

$$\int_{-\infty}^{+\infty} \cdots \int \left\{ (\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)]}{\partial \theta} \right\} f_{\mathbf{x};\theta}(\mathbf{x};\theta)\, d\mathbf{x} = 1 \tag{2.4.14}$$

Clearly, the left side of (2.3.14) is simply the expectation of the expression inside the brackets, that is ,

$$E\{(\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)]}{\partial \theta}\} = 1 \tag{2.4.15}$$

Using the *Cauchy-Schwarz inequality* (Papoulis 1991; Stark and Woods 1994) $| E\{x(\xi)y(\xi)\} |^2 \leq E\{| x(\xi) |^2\}E\{| y(\xi) |^2\}$,we obtain

$$E\{(\hat{\theta} - \theta)^2\} \left\{ \left(\frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)]}{\partial \theta} \right)^2 \right\} \geq E^2 \left\{ (\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)]}{\partial \theta} \right\} = 1 \tag{2.4.16}$$

The first term on the left-hand side is the variance of the estimator $\hat{\theta}$ since it is unbiased. Hence

$$\text{var}(\hat{\theta}) \geq \frac{1}{E\left\{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x};\theta)/\partial\theta]^2\right\}} \tag{2.4.17}$$

which is on e form of the CRLB and can also be expressed as

$$\text{var}(\hat{\theta}) \geq \frac{1}{E\left\{\partial^2 \ln f_{\mathbf{x};\theta}(\mathbf{x};\theta)/\partial\theta^2\right\}} \tag{2.4.18}$$

The function in $f_{\mathbf{x};\theta}(\mathbf{x};\theta)$ is called the *log likelihood function* of θ. The CRLB expresses the minimum error variance of any estimator $\hat{\theta}$ of θ in terms of the joint density $f_{\mathbf{x};\theta}(\mathbf{x};\theta)$ of observations. Hence every unbiased estimator must have a variance greater than a certain number. An unbiased estimate that satisfies the CRLB (2.4.18) with equality is called an efficient estimate. If such an estimate exists, then it can be obtained as a unique solution to the likelihood equation

$$\frac{\partial \ln f_{\mathbf{x};\theta}(\mathbf{x};\theta)}{\partial \theta} = 0 \tag{2.4.19}$$

The solution of (2.4.19) is called the maximum likelihood (ML) estimate. Note that if the efficient estimate does not exist, then the ML estimate will not achieve the lower bound and hence it is difficult to ascertain how closely the variance of any estimate will approach the bound. The CRLB can be generalized to handle the estimation of vector parameters (Therrien 1992).

Consistency of estimator. If the MSE of the estimator can be made to approach zero as the sample size N becomes large, then from (2.4.7) both the bias and the variance will tend to zero. Then the sampling distribution will tend to concentrate about θ, and eventually as $N \to \infty$, the sampling distribution will become an impulse at θ. This is an important and desirable property, and the estimator that possesses it is called a *consistent* estimator.

Confidence interval. If we know the sampling distribution of an estimator, we can use the observations to compute an interval that has a specified probability of covering the unknown true parameter value. This interval is called a ***confidence interval***, and the coverage probability is called the ***confidence level***. When we interpret the meaning of confidence intervals, it is important to remember that it is the interval that is the random variable, and not the parameter. This concept will be explained in the sequel by means of specific examples.

2.4.2 Estimation of Mean

The natural estimator of the mean μ_x of a stationary sequence $x(n)$ from the observations $\{x(n)\}_0^{N-1}$ is the *sample mean*, given by

$$\hat{\mu}_x = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \tag{2.4.20}$$

The estimate $\hat{\mu}_x$ is a random variable that depends on the number and values of the observations. Changing N or the set of observations will lead to another value for $\hat{\mu}_x$. Since the mean of the estimator is given by

$$E\{\hat{\mu}_x\} = \mu_x \tag{2.4.21}$$

the estimator $\hat{\mu}_x$ is unbiased. If $x(n) \sim WN(\mu_x, \sigma_x^2)$, we have

$$var(\hat{\mu}_x) = \frac{\sigma_x^2}{N} \tag{2.4.22}$$

because the samples of the process are uncorrelated random variables. This variance, which is a measure of the estimator's quality, increases if $x(n)$ is nonwhite.

Indeed, for a correlated random sequence, the variance of $\hat{\mu}_x$ is given by (see Problem 2.30)

$$var(\hat{\mu}_x) = N^{-1} \sum_{l=-N}^{N} \left(1 - \frac{|l|}{N}\right) \gamma_x(l) \le N^{-1} \sum_{l=-N}^{N} |\gamma_x(l)| \tag{2.4.23}$$

where $\gamma_x(l)$ is the covariance sequence of $x(n)$. If $\gamma_x(l) \to 0$ as $l \to \infty$, then $var(\hat{\mu}_x) \to 0$ as $N \to \infty$ and hence $\hat{\mu}_x$ is a consistent estimator of μ_x. If $\sum_{l=-\infty}^{\infty} |\gamma_x(l)| < \infty$, then from (2.4.23)

$$\lim_{N\to\infty} N var(\hat{\mu}_x) = \lim_{N\to\infty} \sum_{l=-N}^{N} \left(1 - \frac{|l|}{N}\right) \gamma_x(l) = \sum_{l=-\infty}^{\infty} \gamma_x(l) \tag{2.4.24}$$

The expression for $var(\hat{\mu}_x)$ in (2.4.23) can also be put in the form (see Problem 2.30)

$$var(\hat{\mu}_x) = \frac{\sigma_x^2}{N}[1 + \Delta_N(\rho_x)] \tag{2.4.25}$$

where

$$\Delta_N(\rho_x) = 2\sum_{l=0}^{N} \left(1 - \frac{l}{N}\right) \rho_x(l) \qquad \rho_x(l) = \frac{\gamma_x(l)}{\sigma_x^2} \tag{2.4.26}$$

Since $\Delta_N(\rho_x) \ge 0$, the variance of the estimator increases as the amount of correlation among the samples of $x(n)$ increases. This implies that as the correlation increases, we need more samples to retain the quality of the estimate because each additional sample carries "less information." For this reason the estimation of long-memory processes and processes with infinite variance is extremely difficult.

Sampling distribution. If we know the joint pdf of the random variables $\{x(n)\}_0^{N-1}$, we can determine, at least in principle, the pdf of $\hat{\mu}_x$. For example, if it is assumed that the observations are IID as $\mathcal{N}(\mu_x, \sigma_x^2)$ then from (2.4.21) and (2.4.23), it can be seen that $\hat{\mu}_x$ is normal with mean μ_x and variance σ_x^2/N, that is,

$$f_{\hat{\mu}_x}(\hat{\mu}_x) = \frac{1}{\sqrt{2\pi}(\sigma_x/\sqrt{N})} \exp\left[-\frac{1}{2}\left(\frac{\hat{\mu}_x - \mu_x}{\sigma_x/\sqrt{N}}\right)^2\right] \tag{2.4.27}$$

which is the sampling distribution of the mean. If N is large, then from the central limit theorem, the sampling distribution of the sample mean (2.4.27) is usually very close to the normal distribution, even if the individual distributions are not normal.

If we know the standard deviation σ_x, we can compute the probability

$$\Pr\left\{\mu_x - k\frac{\sigma_x}{\sqrt{N}} < \hat{\mu}_x < \mu_x + k\frac{\sigma_x}{\sqrt{N}}\right\} \tag{2.4.28}$$

that the random variable $\hat{\mu}_x$ is within a certain interval specified by two fixed quantities. A simple rearrangement of the above inequality leads to

$$\Pr\left\{\hat{\mu}_x - k\frac{\sigma_x}{\sqrt{N}} < \mu_x < \hat{\mu}_x + k\frac{\sigma_x}{\sqrt{N}}\right\} \tag{2.4.29}$$

which gives the probability that the fixed quantity μ_x lies between the two random variables $\hat{\mu}_x - k\sigma_x/\sqrt{N}$ and

$\hat{\mu}_x + k\sigma_x/\sqrt{N}$. Hence (2.4.29) provides the probability that an interval with fixed length $2k\sigma_x/\sqrt{N}$ and randomly centered at the estimated mean includes the true mean. If we choose k so that the probability defined by (2.4.29) is equal to 0.95, the interval is known as the 95 percent confidence interval. To understand the meaning of this reasoning, we stress that for each set of measurements we compute a confidence interval that either contains or does not contain the true mean. However, if we repeat this process for a large number of observation sets, about 95 percent of the obtained confidence intervals will include the true mean. We stress that by no means does this imply that a confidence interval includes the true mean with probability 0.95.

If the variance σ_x^2 is unknown, then it has to be determined from the observations. This results in two modifications of (2.4.29). First, σ_x is replaced by

$$\hat{\sigma}_x^2 = \frac{1}{N-1}\sum_{n=0}^{N-1}[x(n)-\hat{\mu}_x]^2 \tag{2.4.30}$$

which implies that the center and the length of the confidence interval are different for each set of observations. Second, the random variable $(\hat{\mu}_x - \mu_x)/(\hat{\sigma}_x/\sqrt{N})$ is distributed according to *Student's t distribution with* $v = N-1$ *degrees of freedom* (Parzen 1960), which tends to a Gaussian for large values of N. In these cases, the factor k in (2.4.29) is replaced by the appropriate value t of Student's distribution, using $N-1$ degrees of freedom, for the desired level of confidence.

If the observations are normal but not IID, then from (2.4.25), the mean estimator $\hat{\mu}_x$ is normal with mean μ and variance $(\sigma_x^2/N)[1+\Delta_N(\rho_x)]$. It is now easy to construct exact confidence intervals for $\hat{\mu}_x$ if $\rho_x(l)$ is known, and approximate confidence intervals if $\rho_x(l)$ is to be estimated from the observations. For large N, the variance $var(\hat{\mu}_x)$ can be approximated by

$$\begin{aligned} var(\hat{\mu}_x) &= \frac{\sigma_x^2}{N}[1+\Delta_N(\rho_x)] \simeq \frac{\sigma_x^2}{N}\left[1+2\sum_1^N \rho_x(l)\right] \triangleq \frac{v}{N} \\ v &= \sigma_x^2\left\{1+2\sum_1^N \rho_x(l)\right\} \end{aligned} \tag{2.4.31}$$

and hence an approximate 95 percent confidence interval for $\hat{\mu}_x$ is given by

$$\left(\hat{\mu}_x - 1.96\sqrt{\frac{v}{N}},\ \hat{\mu}_x + 1.96\sqrt{v/N}\right) \tag{2.4.32}$$

This means that, on average, the above interval will enclose the true value μ_x on 95 percent of occasions. For many practical random processes (especially those modeled as ARMA processes), the result in (2.4.32) is a good approximation.

EXAMPLE 2.4.1. Consider the AR(1) process

$$x(n) = ax(n-1) + \omega(n) \qquad -1 < a < 1$$

where $\omega(n) \sim \text{WN}(0, \sigma_\omega^2)$. We wish to compute the variance of the mean estimator $\hat{\mu}_x$ of the process $x(n)$. Using straightforward calculations, we obtain

$$\mu_x = 0 \qquad \sigma_x^2 = \frac{\sigma_\omega^2}{1-a^2} \qquad \text{and} \qquad \rho_x(l) = a^{|l|}$$

From (2.4.26) we evaluate the term

$$\Delta_N(\rho) = \frac{2a}{1-a}\left[1 - \frac{1}{N(1-a)} + \frac{a^N}{N(1-a)}\right] \simeq \frac{2a}{1-a} \qquad \text{for } N \gg 1$$

When $a \to 1$, that is, when the dependence between the signal samples increases, then the factor $\Delta_N(\rho)$ takes large values and the quality of estimator decreases drastically. Similar conclusions can be drawn using the approximation (2.4.31)

$$v = \left(1+2\sum_1^\infty a^l\right)\frac{\sigma_\omega^2}{1-a^2} = \frac{\sigma_\omega^2}{(1-a)^2}$$

We will next verify these results using two Monte Carlo simulations: one for $a = 0.9$, which represents high correlations among samples, and the other for $a = 0.1$. Using a Gaussian pseudorandom number generator with mean 0 and variance $\sigma_\omega^2 = 1$, we generated $N = 100$ samples of the AR(1) process $x(n)$. Using v in (2.4.31) and (2.4.32), we next computed the confidence

intervals. For $a = 0.9$, we obtain

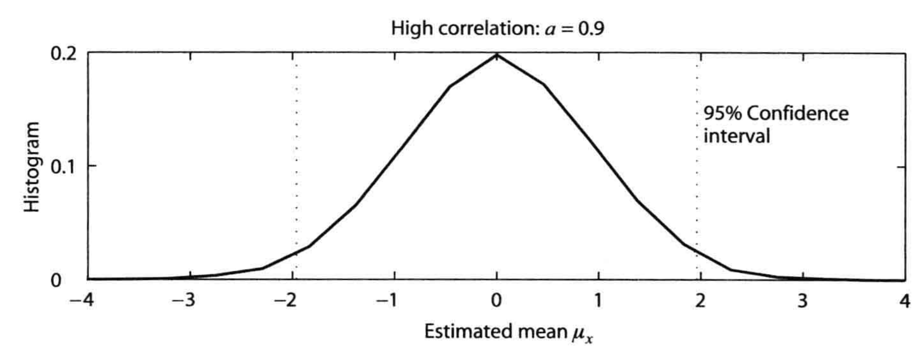
$$v = 100 \quad \text{and} \quad \text{confidence interval}: (\hat{\mu}_x - 0.98,\ \hat{\mu}_x + 0.98)$$

and for $a = 0.1$, we obtain

$$v = 1.2345 \quad \text{and} \quad \text{confidence interval}: (\hat{\mu}_x - 0.2178,\ \hat{\mu}_x + 0.2178)$$

Clearly, when the dependence between signal samples increases, the quality of the estimator decreases drastically and hence the confidence interval is wider. To have the same confidence interval, we should increase the number of samples N.

We next estimate the mean, using (2.4.20), and we repeat the experiment 10,000 times. Figure 2.8 shows histograms of the computed means for $a = 0.9$ and $a = 0.1$. The confidence intervals are also shown as dotted lines around the true mean. The histograms are approximately Gaussian in shape. The histogram for the high-correlation case is wider than that for the low-correlation case, which is to be expected. The 95 percent confidence intervals also indicate that very few estimates are outside the interval.



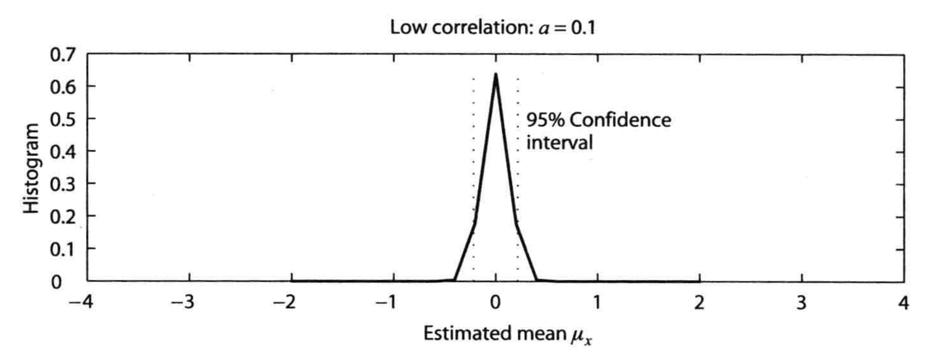


FIGURE 2.8
Histograms of mean estimates in Example 2.4.1.

2.4.3 Estimation of Variance

The natural estimator of the variance σ_x of a stationary sequence $x(n)$ from the observations $\{x(n)\}_0^{N-1}$ is the ***sample variance***, given by

$$\hat{\sigma}_x^2 \triangleq \frac{1}{N} \sum_{n=0}^{N-1} \{x(n) - \hat{\mu}_x\}^2 \tag{2.4.33}$$

By using the mean estimate $\hat{\mu}_x$ from (2.4.20), the mean of the variance estimator can be shown to equal (see Problem 2.31)

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - var(\hat{\mu}_x) = \sigma_x^2 - \frac{1}{N} \sum_{l=-N}^{N} \left(1 - \frac{|l|}{N}\right) \gamma_x(l) \tag{2.4.34}$$

If the sequence $x(n)$ is uncorrelated, then

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - \frac{\sigma_x^2}{N} = \left(\frac{N-1}{N}\right) \sigma_x^2 \tag{2.4.35}$$

From (2.4.34) or (2.4.35), it is obvious that the estimator in (2.4.33) is biased. If $\gamma_x(l) \to 0$ as $l \to \infty$, then $var(\hat{\mu}_x) \to 0$ as $N \to \infty$ and hence $\hat{\sigma}_x^2$ is an asymptotically unbiased estimator of σ_x^2. In practical applications, the variance estimate is nearly unbiased for large N. Note that if we use the actual mean μ_x in (2.4.33), then the

resulting estimator is unbiased.

The general expression for the variance of the variance estimator is fairly complicated and requires higher-order moments. It can be shown that for either estimators

$$var(\hat{\sigma}_x^2) \approx \frac{\gamma_x^{(4)}}{N} \qquad \text{for large } N \tag{2.4.36}$$

where $\gamma_x^{(4)}$ is the fourth central moment of $x(n)$ (Brockwell and Davis 1991). Thus the estimator in (2.6.33) is also consistent.

Sampling distribution. In the case of the mean estimator, the sampling distribution involved the distribution of sums of random variables. The variance estimator involves the sum of the squares of random variables, for which the sampling distribution computation is complicated. For example, if there are N independent measurements from an $N(0,1)$ distribution, then the sampling distribution of the random variable

$$\chi_N^2 = x_1^2 + x_2^2 + \cdots + x_N^2 \tag{2.4.37}$$

is given by the *chi-squared distribution with N degrees of freedom.* The general form of χ_N^2 with ν degrees of freedom is

$$f_{\chi_\nu^2}(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} x^{\nu/2-1} \exp\left(-\frac{x}{2}\right) \qquad 0 \le x \le \infty \tag{2.4.38}$$

where $\Gamma(\nu/2) = \int_0^\infty e^{-t} t^{\nu/2-1}\, dt$ is the gamma function with argument $\nu/2$.

For the variance estimator in (2.4.33), it can be shown (Parzen 1960) that $N\hat{\sigma}_x^2$ is distributed as chi squared with $\nu = N-1$ *degrees of freedom*. This means that, for any set of N observations, there will only be $N-1$ independent deviations $\{x(n) - \hat{\mu}_x\}$, since their sum is zero from the definition of the mean. Assuming that the observations are $\mathcal{N}(\mu, \sigma^2)$, the random variables $x(n)/\sigma$ will be $\mathcal{N}(\mu/\sigma,\ 1)$ and hence the random variable

$$\frac{N\hat{\sigma}_x^2}{\sigma^2} = \frac{1}{\sigma^2}\sum_{n=0}^{N-1}[x(n) - \hat{\mu}_x]^2 \tag{2.4.39}$$

will be chi squared distributed with $\nu = N-1$. Therefore, using values of the chi-squared distribution, confidence intervals for the variance estimator can be computed. In particular, since $N\hat{\sigma}_x^2/\sigma^2$ is distributed as χ_ν^2, the 95 percent limits of the form

$$\Pr\left\{\chi_\nu\left(\frac{0.05}{2}\right) < N\hat{\sigma}_x^2/\sigma^2 \le \chi_\nu\left(1 - \frac{0.05}{2}\right)\right\} = 0.95 \tag{2.4.40}$$

can be obtained from chi-squared tables (Fisher and Yates 1938). By rearranging (2.4.40), the random variable $\sigma^2/\hat{\sigma}_x^2$ satisfies

$$\Pr\left\{\frac{N}{\chi_\nu(0.975)} < \frac{\sigma^2}{\hat{\sigma}_x^2} \le \frac{N}{\chi_\nu(0.025)}\right\} = 0.95 \tag{2.4.41}$$

Using $l_1 = N/\chi_\nu(0.975)$ and $l_2 = N/\chi_\nu(0.025)$, we see that (2.4.41) implies that

$$\Pr\{l_2\hat{\sigma}_x^2 \ge \sigma^2 \quad \text{and} \quad l_1\hat{\sigma}_x^2 < \sigma^2\} = 0.95 \tag{2.4.42}$$

Thus the 95 percent confidence interval based on the estimate $\hat{\sigma}_x^2$ is $(l_1\hat{\sigma}_x^2, l_2\hat{\sigma}_x^2)$. Note that this interval is sensitive to the validity of the normal assumption of random variables leading to (2.4.39). This is not the case for the confidence intervals for the mean estimates because, thanks to the central limit theorem, the computation of the interval can be based on the normal assumption.

EXAMPLE 2.4.2 Consider again the AR(1) process given in Example 2.4.1:

$$x(n) = ax(n-1) + \omega(n) \qquad -1 < a < 1 \qquad \omega(n) \sim \text{WN}(0,1)$$

with

$$\mu_x = 0 \qquad \sigma_x^2 = \frac{\sigma_\omega^2}{1-a^2} \qquad \text{and} \qquad \rho_x(l) = a^{|l|} \tag{2.4.43}$$

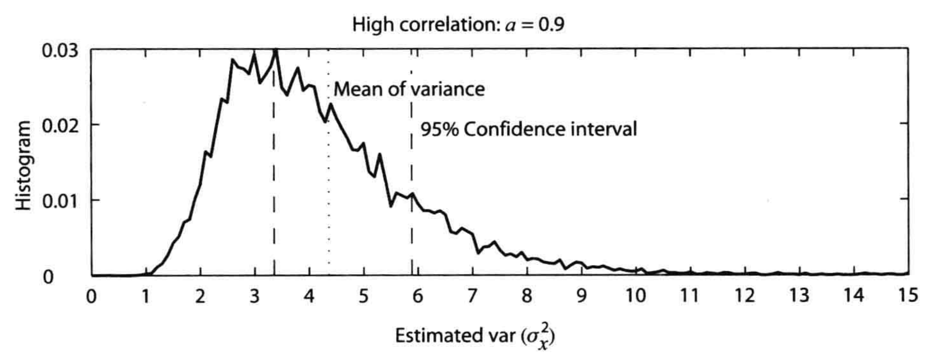
We wish to compute the mean of the variance estimator $\hat{\sigma}_x^2$ of the process $x(n)$. From (2.4.34), we obtain

$$E[\hat{\sigma}_x^2] = \sigma_x^2 \left[1 - \frac{1}{N}\sum_{l=-N}^{N}\left(1 - \frac{|l|}{N}\right)a^{|l|}\right] \quad (2.4.44)$$

When $a \to 1$, that is, when the dependence between the signal samples increases, the mean of the estimate deviates significantly from the true value σ_x^2 and the quality of the estimator decreases drastically. For small dependence, the mean is very close to σ_x^2. These conclusions can be verified using two Monte Carlo simulations as before: one for $a = 0.9$, which represents high correlations among samples, and the other for $a = 0.1$. Using a Gaussian pseudorandom number generator with mean 0 and unit variance, we generated $N = 100$ samples of the AR(1) process $x(n)$. The computed parameters according to (2.4.43) and (2.4.44) are

$$a = 0.9: \quad \sigma_x^2 = 5.2632 \quad E\{\hat{\sigma}_x^2\} = 4.3579$$
$$a = 0.1: \quad \sigma_x^2 = 1.0101 \quad E\{\hat{\sigma}_x^2\} = 0.9978$$

We next estimate the variance by using (2.4.33) and repeat the experiment 10,000 times. Figure 2.9 shows histograms of computed variances for $a = 0.9$ and for $a = 0.1$. The computed means of the variance estimates are also shown as dotted lines. Clearly, the histogram is much wider for the high-correlation case and much narrower (almost symmetric and Gaussian) for the low-correlation case.



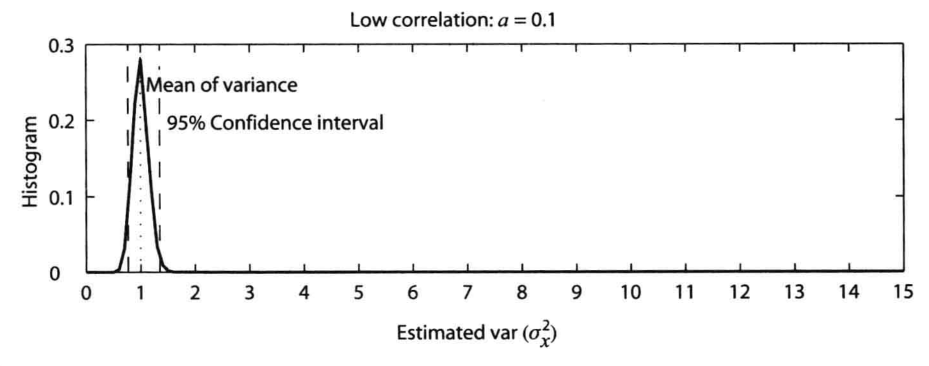


FIGURE 2.9

Histograms of variance estimates in Example 3.4.2.

The 95 percent confidence intervals are given by $(l_1\hat{\sigma}_x^2,\ l_2\hat{\sigma}_x^2)$, where $l_1 = N/\chi_v(0.975)$ and $l_2 = N/\chi_v(0.025)$. The values of l_1 and l_2 are obtained from the chi-squared distribution curves (Jenkins and Watts 1968). For $N = 100$, $l_1 = 0.77$ and $l_2 = 1.35$; hence the 95 percent confidence intervals for σ_x^2 are

$$(0.77\hat{\sigma}_x^2,\ 1.35\hat{\sigma}_x^2)$$

also shown as dashed lines around the mean value $E\{\hat{\sigma}_x^2\}$. The confidence interval for the high-correlation case, $a = 0.9$, does not appear to be a good interval, which implies that the approximation leading to (2.4.42) is not a good one for this case. Such is not the case for $a = 0.1$.

2.5 Summary

In this chapter we provided an overview of the basic theory of discrete-time stochastic processes. We began with the descriptron of stochastic processes. To describe stochastic processes, we proceeded to define mean and autocorrelation sequences. In many applications, the concept of stationary of random processes is a useful one that

reduces the computational complexity. Assuming time invariance on the first two moments, we defined a wide-sense stationary (WSS) process in which the mean is a constant and correlation between random variables at two distinct times is a function of time difference or lag. The rest of the chapter was devoted to the analysis of WSS processes.

A stochastic process is generally observed in practice as a single sample function (a speech signal or a radar signal) from which it is necessary to estimate the first- and the second-order moments. This requires the notion of ergodicity, which provides a framework for the computation of statistical averages using time averages over a single realization. Although this framework requires theoretical results using mean square convergence, we provided a simple approach of using appropriate time averages. An important random signal characteristic called variability was introduced. The WSS processes were then described in the frequency domain using the power spectral density function, which is a physical quantity that can be measured in practice. Some random processes exhibiting flat spectral envelopes were analyzed including one of white noise. Since random processes are generally processed using linear systems, we described linear system operations with random inputs in both the time and frequency domains.

The properties of correlation matrices and sequences play an important role in filtering and estimation theory and were discussed in detail, including eigenanalysis. Another important random signal characteristic called memory was also introduced. Stationary random signals were modeled using autocorrelation matrices, and the relationship between spectral flatness and eigenvalue spread was explored. These properties were used in an alternate representation of random vectors as well as processes using uncorrelated components which were based on diagonalization and triangularization of correlation matrices.

Finally, we concluded this chapter with the introduction of elementary estimation theory. After discussion of properties of estimators, two important estimators of mean and variance were treated in detail along with their sampling distributions. These topics will be useful in many subsequent chapters.

Problems

2.1 The exponential density function is given by

$$f_x(x) = \frac{1}{a} e^{-x/a} u(x) \tag{P.1}$$

where a is a parameter and $u(x)$ is a unit step function.

(*a*) Plot the density function for $a = 1$.

(*b*) Determine the mean, variance, skewness, and kurtosis of the Rayleigh random variable with $a = 1$. Comment on the significance of these moments in terms of the shape of the density function.

(*c*) Determine the characteristic function of the exponential pdf.

2.2 The Rayleigh density function is given by

$$f_x(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} u(x) \tag{P.2}$$

where σ is a parameter and $u(x)$ is a unit step function. Repeat Problem 2.1 for $\sigma = 1$.

2.3 Using the binomial expansion of $\{x(\zeta) - \mu_x\}^m$, show that the m th central moment is given by

$$M_m^{(x)} = \sum_{k=0}^{m} \binom{m}{k} (-1)^k \mu_x^k \xi_{m-k}^{(x)}$$

Similarly, show that

$$\xi_m^{(x)} = \sum_{k=0}^{m} \binom{m}{k} \mu_x^k M_{m-k}^{(x)}$$

2.4 Consider a zero-mean random variable $x(\zeta)$. Let us discuss *cumulant generating function* and is given by

$$\bar{\Psi}_x(s) \triangleq \ln \bar{\Phi}_x(s) = \ln E\{e^{sx(\zeta)}\}$$

When s is replaced by $j\xi$ in (P.2.1), the resulting function is known as the *second characteristic function* and is denoted by $\Psi_x(\xi)$.

The *cumulants* $\kappa_x^{(m)}$ of a random variable $x(\zeta)$ are defined as the derivatives of the cumulant generating function, that is,

$$\kappa_x^{(m)} \triangleq \left. \frac{d^m[\bar{\Psi}_x(s)]}{ds^m} \right|_{s=0} = (-j)^m \left. \frac{d^m[\Psi_x(\xi)]}{d\xi^m} \right|_{\xi=0} \qquad m = 1, 2, \cdots$$

Clearly, $\kappa_x^{(0)} = 0$. It can be shown that for a zero-mean random variable, the first five cumulants as functions of the central

moments are given by

$$\kappa_x^{(1)} = r_1^{(x)} = \mu_x = 0$$
$$\kappa_x^{(2)} = \gamma_x^{(2)} = \sigma_x^2$$
$$\kappa_x^{(3)} = \gamma_x^{(3)}$$
$$\kappa_x^{(4)} = \gamma_x^{(4)} - 3\sigma_x^4$$

Using (P.2.1), show that the first four cumulants of $x(\zeta)$ are given by (P.2.3) through (P.2.6).

2.5 A random vector $\mathbf{x}(\zeta) = [x_1(\zeta)\ x_2(\zeta)]^T$ has mean vector $\boldsymbol{\mu}_\mathbf{x} = [1\ 2]^T$ and covariance matrix

$$\Gamma_\mathbf{x} = \begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

This vector is transformed to another random vector $\mathbf{y}(\zeta)$ by the following linear transformation:

$$\begin{bmatrix} y_1(\zeta) \\ y_2(\zeta) \\ y_3(\zeta) \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_1(\zeta) \\ x_2(\zeta) \end{bmatrix}$$

Determine (*a*) the mean vector $\boldsymbol{\mu}_\mathbf{y}$, (*b*) the autocovariance matrix $\Gamma_\mathbf{y}$, and (*c*) the cross-correlation matrix $\boldsymbol{R}_{xy}$.

2.6 Using the moment generating function, show that the linear transformation of a Gaussian random vector is also Gaussian.

2.7 Let $\{x_k(\zeta)\}_{k=1}^4$ be four IID random variables with exponential distribution (P.1) with $a = 1$. Let

$$y_k(\zeta) = \sum_{l=1}^{k} x_l(\zeta) \qquad 1 \le k \le 4$$

(*a*) Determine and plot the pdf of $y_2(\zeta)$.
(*b*) Determine and plot the pdf of $y_3(\zeta)$.
(*c*) Determine and plot the pdf of $y_4(\zeta)$.
(*d*) Compare the pdf of $y_4(\zeta)$ with that of the Gaussian density.

2.8 For each of the following, determine whether the random process is (1) WSS or (2) m.s. ergodic in the mean.
(*a*) $X(t) = A$, where *A* is a random variable uniformly distributed between 0 and 1.
(*b*) $X_n = A\cos\omega_0 n$, where *A* is a Gaussian random variable with mean 0 and variance 1.
(*c*) A Bernoulli process with $\Pr[X_n = 1] = p$ and $\Pr[X_n = -1] = 1 - p$.

2.9 Consider the harmonic process $x(n)$ defined in (2.1.50).
(*a*) Determine the mean of $x(n)$.
(*b*) Show that the autocorrelation sequence is given by

$$r_x(l) = \frac{1}{2}\sum_{k=1}^{N} |c_k|^2 \cos\omega_k l \qquad -\infty < l < \infty$$

2.10 Suppose that the random variables φ_k in the real-valued harmonic process model are distributed with a pdf $f_{\varphi_k}(\varphi_k) = (1 + \cos\varphi_k)/(2\pi),\ -\pi \le \varphi_k \le \pi$. Is the resulting stochastic process stationary?

2.11 A stationary random sequence $x(n)$ with mean $\mu_x = 4$ and autocovariance

$$\gamma_x(n) = \begin{cases} 4 - |n| & |n| \le 3 \\ 0 & \text{otherwise} \end{cases}$$

is applied as an input to a linear shift-invariant (LSI) system whose impulse response $h(n)$ is

$$h(n) = u(n) - u(n-4)$$

where $u(n)$ is a unit step sequence. The output of this system is another random sequence $y(n)$. Determine (*a*) the mean sequence $\mu_y(n)$, (*b*) the cross-covariance $\gamma_{xy}(n_1, n_2)$ between $x(n_1)$ and $y(n_2)$, and (*c*) the autocovariance $\gamma_y(n_1, n_2)$ of the output process $y(n)$.

2.12 A causal LTI system, which is described by the difference equation

$$y(n) = \frac{1}{2}y(n-1) + x(n) + \frac{1}{3}x(n-1)$$

is driven by a zero-mean WSS process with autocorrelation $r_x(l) = 0.5^{|l|}$.
(*a*) Determine the PSD and the autocorrelation of the output sequence $y(n)$.
(*b*) Determine the cross-correlation $r_{xy}(l)$ and cross-PSD $R_{xy}(e^{j\omega})$ between the input and output signals.

2.13 A WSS process with PSD $R_x(e^{j\omega}) = 1/(1.64 + 1.6\cos\omega)$ is applied to a causal system described by the following difference

equation

$$y(n) = 0.6y(n-1) + x(n) + 1.25x(n-1)$$

Compute (*a*) the PSD of the output and (*b*) the cross-PSD $R_{xy}(e^{j\omega})$ between input and output.

2.14 Determine whether the following matrices are valid correlation matrices:

$$\text{(a)}\boldsymbol{R}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \text{(b)}\boldsymbol{R}_2 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix}$$

$$\text{(c)}\boldsymbol{R}_3 = \begin{bmatrix} 1 & 1-j \\ 1+j & 1 \end{bmatrix} \qquad \text{(d)}\boldsymbol{R}_4 = \begin{bmatrix} 1 & \frac{1}{2} & 1 \\ \frac{1}{2} & 2 & \frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

2.15 Consider a normal random vector $\mathbf{x}(\zeta)$ with components that are mutually uncorrelated, that is, $\rho_{ij} = 0$. Show that (*a*) the covariance matrix $\Gamma_{\mathbf{x}}$ is diagonal and (*b*) the components of $\mathbf{x}(\zeta)$ are mutually independent.

2.16 Show that if a real, symmetric, and nonnegative definite matrix $\boldsymbol{R}$ has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$, then the matrix $\boldsymbol{R}^k$ has eigenvalues $\lambda_1^k, \lambda_2^k, \ldots, \lambda_M^k$.

2.17 Prove that the trace of $\boldsymbol{R}$ is given by

$$\text{tr}\boldsymbol{R} = \sum \lambda_i$$

2.18 Prove that the determinant of $\boldsymbol{R}$ is given by

$$\det \boldsymbol{R} = |\boldsymbol{R}| = \prod \lambda_i = |\Lambda|$$

2.19 Show that the determinants of $\boldsymbol{R}$ and Γ are related by

$$\det \boldsymbol{R} = \det \Gamma(1 + \boldsymbol{\mu}^H \boldsymbol{\Gamma} \boldsymbol{\mu})$$

2.20 Let $\mathbf{R}_{\mathbf{x}}$ be the correlation matrix of the vector $\boldsymbol{x} = [x(0)\ \ x(2)\ \ x(3)]^T$, where $x(n)$ is a zero-mean WSS process.

(*a*) Check whether the matrix $\boldsymbol{R}_x$ is Hermitian, Toeplitz, and nonnegative definite.

(*b*) If we know the matrix $\boldsymbol{R}_x$, can we determine the correlation matrix of the vector $\overline{\boldsymbol{x}} = [x(0)\ \ x(1)\ \ x(2)\ \ x(3)]^T$?

2.21 Using the nonnegativeness of $E\{[x(n+l) \pm x(n)]^2\}$, show that $r_x(0) \geq |r_x(l)|$ for all *l*.

2.22 Show that $r_x(l)$ is nonnegative definite, that is,

$$\sum_{l=1}^{M}\sum_{k=1}^{M} a_l r_x(l-k) a_k^* \geq 0 \qquad \forall M, \forall a_1, \cdots, a_M$$

2.23 Let $x(n)$ be a random process generated by the AP(1) system

$$x(n) = \alpha x(n-1) + \omega(n) \qquad n \geq 0 \qquad x(-1) = 0$$

where $\omega(n)$ is an $IID(0, \sigma_\omega^2)$ process.

(*a*) Determine the autocorrelation $r_x(n_1, n_2)$ function.

(*b*) Show that $r_x(n_1, n_2)$ asymptotically approaches $r_x(n_1 - n_2)$, that is, it becomes shift-invariant.

2.24 Let $\mathbf{x}$ be a random vector with mean $\mu_{\mathbf{x}}$ and autocorrelation $\boldsymbol{R}_x$.

(*a*) Show that $\boldsymbol{y} = \boldsymbol{Q}^T \boldsymbol{x}$ transforms $\boldsymbol{x}$ to an uncorrelated component vector y if $\boldsymbol{Q}$ is the eigenmatrix of $\boldsymbol{R}_x$.

(*b*) Comment on the geometric interpretation of this transformation.

2.25 The mean and the covariance of a Gaussian random vector $\boldsymbol{x}$ are given by, respectively,

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Gamma}_{\mathbf{x}} = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$$

Plot the $1\sigma, 2\sigma,$ and 3σ concentration ellipses representing the contours of the density function in the (x_1, x_2) plane. *Hints:* The radius of an ellipse with major axis a (along x_1) and minor axis $b<a$ (along x_2) is given by

$$r^2 = \frac{a^2b^2}{a^2\sin^2\theta + b^2\cos^2\theta}$$

where $0 \le \theta \le 2\pi$. Compute the 1σ ellipse specified by $a=\sqrt{\lambda_1}$ and $b=\sqrt{\lambda_2}$ and then rotate and translate each point $\boldsymbol{x}^{(i)} = [x_1^{(i)}\ x_2^{(i)}]^{\mathrm{T}}$ using the transformation $\boldsymbol{w}^{(i)} = \boldsymbol{Q}_x\boldsymbol{x}^{(i)} + \mu_x$.

2.26 Consider the process $x(\mu) = ax(\mu-1) + \omega(\mu)$, where $\omega(\mu) \sim \mathrm{WN}(0, \sigma_\omega^2)$.

(*a*) Show that the $M\times M$ correlation matrix of the process is symmetric Toeplitz and is given by

$$\boldsymbol{R}_x = \frac{\sigma_\omega^2}{1-a^2}\begin{bmatrix} 1 & a & \cdots & a^{m-1} \\ a & 1 & \cdots & a^{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{m-1} & a^{m-2} & \cdots & 1 \end{bmatrix}$$

(*b*) Verify that

$$\boldsymbol{R}_x^{-1} = \frac{1}{\sigma_\omega^2}\begin{bmatrix} 1 & -a & 0 & \cdots & 0 \\ -a & 1+a^2 & -a & \cdots & 0 \\ 0 & -a & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & 1+a^2 & -a \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix}$$

(*c*) Show that if

$$\boldsymbol{L}_x = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -a & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & -a & 1 \end{bmatrix}$$

then $\boldsymbol{L}_x^{\mathrm{T}}\boldsymbol{R}_x\boldsymbol{L}_x = (1-a^2)\boldsymbol{I}$.

(*d*) For $\sigma_\omega^2 = 1, a = 0.95$, and $M=8$ compute the DKLT and the DFT.

(*e*) Plot the eigenvalues of each transform in the same graph of the PSD of the process. Explain your findings.

(*f*) Plot the eigenvectors of each transform and compare the results.

(*g*) Repeat parts (*e*) and (*f*) for $M=16$ and $M=32$. Explain the obtained results.

(*h*) Repeat parts (*e*) to (*g*) for $a=0.5$ and compare with the results obtained for $a=0.95$.

2.27 Determine three different innovations representations of a zero-mean random vector $\boldsymbol{x}$ with correlation matrix

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}$$

2.28 Verify that the eigenvalues and eigenvectors of the $M\times M$ correlation matrix of the process $x(\mu) = \omega(\mu) + b\omega(n-1)$, where $\omega(n) \sim \mathrm{WN}(0,\sigma_\omega^2)$ are given by $\lambda_k = R_x(e^{j\omega_k})$, $q_n^{(k)} = \sin\omega_k n$, $\omega_k = \pi k/(M+1)$, where $k = 1, 2, \cdots, M$, (***a***) analytically and (***b***) numerically for $\sigma_\omega^2 = 1$ and $M=8$. *Hint:* Plot the eigenvalues on the same graph with the PSD.

2.29 Consider the process $x(n) = \omega(n) + b\omega(n-1)$.

(*a*) Compute the DKLT for $M=3$.

(*b*) Show that the variances of the DKLT coefficients are $\sigma_x^2(1+\sqrt{2}b)$, σ_x^2, and $\sigma_x^2(1-\sqrt{2}b)$.

2.30 Let $x(n)$ be a stationary random process with mean μ_x and covariance $\gamma_x(l)$. Let $\hat{\mu}_x = 1/N\sum_{n=0}^{N-1} x(n)$ be the sample mean from the observations $\{x(n)\}_{n=0}^{N-1}$.

(*a*) Show that the variance of $\hat{\mu}_x$ is given by

$$var(\hat{\mu}_x) = N^{-1}\sum_{l=-N}^{N}\left(1-\frac{|l|}{N}\right)\gamma_x(l) \le N^{-1}\sum_{l=-N}^{N}|\gamma_x(l)| \qquad \text{(P.4)}$$

(*b*) Show that the above result (P.4) can be expressed as

$$var(\hat{\mu}_x) = \frac{\sigma_x^2}{N}[1 + \Delta_N(\rho_x)] \tag{P.5}$$

where $$\Delta_N(\rho_x) = 2\sum_{l=0}^{N}\left(1 - \frac{l}{N}\right)\rho_x(l) \qquad \rho_x(l) = \frac{\gamma_x(l)}{\sigma_x^2}$$

(*c*) Show that (P.4) reduces to $var(\hat{\mu}_x) = \sigma_x^2 / N$ for a $\mathrm{WN}(\mu_x, \sigma_x^2)$ process.

2.31 Let $x(n)$ be a stationary random process with mean μ_x, variance σ_x^2, and covariance $\gamma_x(l)$. Let

$$\hat{\sigma}_x^2 \triangleq \frac{1}{N}\sum_{n=0}^{N-1}[x(n) - \hat{\mu}_x]^2$$

be the sample variance from the observations $\{x(n)\}_{n=0}^{N-1}$.

(*a*) Show that the mean of $\hat{\sigma}_x^2$ is given by

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - var(\hat{\mu}_x) = \sigma_x^2 - \frac{1}{N}\sum_{l=-N}^{N}\left(1 - \frac{|l|}{N}\right)\gamma_x(l)$$

(*b*) Show that the above result reduces to $var(\hat{\mu}_x) = (N-1)\sigma_x^2/N$ for a $\mathrm{WN}(\mu_x, \sigma_x^2)$ process.

2.32 The Cauchy distribution with mean μ is given by

$$f_x(x) = \frac{1}{\pi}\frac{1}{1 + (x - \mu)^2} \qquad -\infty < x < \infty$$

Let $\{x_k(\zeta)\}_{i=k}^{N}$ be N IID random variables with the above distribution. Consider the mean estimator based on $\{x_k(\zeta)\}_{i=k}^{N}$

$$\hat{\mu}(\zeta) = \frac{1}{N}\sum_{k=1}^{N} x_k(\zeta)$$

Determine whether $\hat{\mu}(\zeta)$ is a consistent estimator of μ.

CHAPTER 3

Linear Signal Models

In this chapter we introduce and analyze the properties of a special class of stationary random sequences that are obtained by driving a linear, time-invariant system with white noise. We focus on filters having a system function that is rational, that is, the ratio of two polynomials. The power spectral density of the resulting process is also rational, and its shape is completely determined by the filter coefficients. We will use the term *pole-zero* models when we want to emphasize the system viewpoint and the term *autoregressive moving-average* models to refer to the resulting random sequences. The latter term is not appropriate when the input is a harmonic process or a deterministic signal with a flat spectral envelope. We discuss the impulse response, autocorrelation, power spectrum, partial autocorrelation, and cepstrum of all-pole, all-zero, and pole-zero models. We express all these quantities in terms of the model coefficients and develop procedures to convert from one parameter set to another. Low-order models are studied in detail, because they are easy to analyze analytically and provide insight into the behavior and properties of higher-order models. An understanding of the correlation and spectral properties of a signal model is very important for the selection of the appropriate model in practical applications.

3.1 Introduction

In Chapter 2 we defined and studied random processes as a mathematical tool to analyze random signals. In practice, we also need to generate random signals that possess certain known, second-order characteristics, or we need to describe observed signals in terms of the parameters of known random processes.

The simplest random signal model is the wide sense stationary white noise sequence $\omega(n) \sim \mathrm{WN}(0, \sigma_{\omega}^{2})$ that has uncorrelated samples and a flat PSD. It is also easy to generate in practice by using simple algorithms. If we filter white noise with a stable LTI filter, we can obtain random signals with almost any arbitrary aperiodic correlation structure or continuous PSD. If we wish to generate a random signal with a line PSD using the previous approach, we need an LTI filter with "line" frequency response; that is, we need an oscillator. Unfortunately, such a system is not *stable*, and its output cannot be stationary. Fortunately, random signals with line PSDs can be easily generated by using the *harmonic process* model (linear combination of sinusoidal sequences with statistically independent random phases) discussed in Section 2.1.6 Figure 3.1 illustrates the filtering of white noise and "white " (flat spectral envelope) harmonic process by an LTI filter. Signal models with mixed PSDs can be obtained by combining the above two models, a process justified by a powerful result known as the *Wold decomposition*.

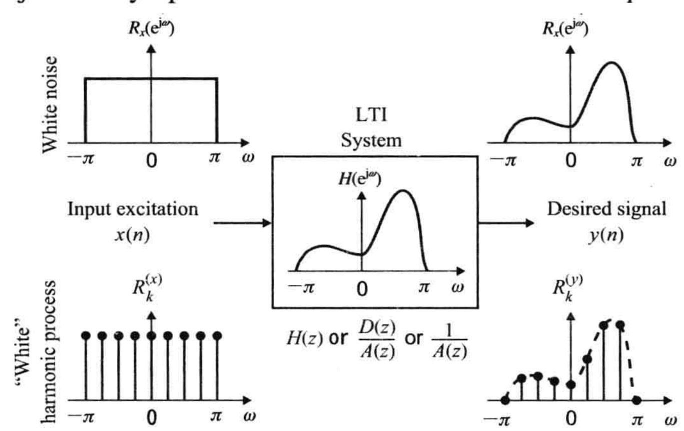


FIGURE 3.1
Signal models with continuous and discrete(line)power spectrum densities.

When the LTI filter is specified by its impulse response, we have a *nonparametric* signal model because there is no restriction regarding the form of the model and the number of parameters is infinite. However, if we specify the

filter by a finite-order rational system function, we have a *parametric* signal model described by a finite number of parameters. We focus on parametric models because they are simpler to deal with in practical applications. The two major topics we address in this chapter are (1) the derivation of the second-order moments of AP, AZ, and PZ models, given the coefficients of their system function, and (2) the design of an AP, AZ, or PZ system that produces a random signal with a given autocorrelation sequence or PSD function. The second problem is known as *signal modeling* and theoretically is equivalent to the spectral factorization procedure developed. The modeling of harmonic processes is theoretically straightforward and does not require the use of a linear filter to change the amplitude of the spectral lines. The challenging problem in this case is the identification of the filter by observing its response to a harmonic process with a flat PSD. The modeling problem for continuous PSDs has a solution, at least in principle, for every regular random sequence.

In practical applications, the second-order moments of the signal to be modeled are not known a priori and have to be estimated from a set of signal observations. This element introduces a new dimension and additional complications to the signal modeling problem, which are discussed in Chapter 8. In this chapter we primarily focus on parametric models that replicate the second-order properties (autocorrelation or PSD) of stationary random sequences. If the sequence is Gaussian, the model provides a complete statistical characterization.

3.1.1 Linear Nonparametric Signal Models

Consider a stable LTI system with impulse response $h(n)$ and input $\omega(n)$. The output $x(n)$ is given by the convolution summation

$$x(n)=\sum_{k=-\infty}^{\infty} h(k)\omega(n-k) \tag{3.1.1}$$

which is known as a *nonrecursive* system representation because the output is computed by linearly weighting samples of the input signal.

Linear random signal model. If the input $\omega(n)$ is a zero-mean white noise process with variance σ_ω^2, autocorrelation $r_\omega(l)=\sigma_\omega^2\delta(l)$, and PSD $R_\omega(e^{j\omega})=\sigma_\omega^2$, $-\pi<\omega\le\pi$, then from Table 2.2 the autocorrelation, complex PSD, and PSD of the output $x(n)$ are given by, respectively,

$$r_x(l)=\sigma_\omega^2\sum_{k=-\infty}^{\infty} h(k)h^*(k-l)=\sigma_\omega^2 r_h(l) \tag{3.1.2}$$

$$R_x(z)=\sigma_\omega^2 H(z)H^*\left(\frac{1}{z^*}\right) \tag{3.1.3}$$

$$R_x(e^{j\omega})=\sigma_\omega^2|H(e^{j\omega})|^2=\sigma_\omega^2 R_h(e^{j\omega}) \tag{3.1.4}$$

We notice that when the input is a white noise process, the shape of the autocorrelation and the power spectrum (*second-order moments*) of the output signal are completely characterized by the system. We use the term *system-based signal model* to refer to the signal generated by a system with a white noise input. If the system is linear, we use the term *linear random signal model*. In the statistical literature, the resulting model is known as the *general linear process model*. However, we should mention that in some applications it is more appropriate to use a deterministic input with flat spectral envelope or a "white" harmonic process input.

Recursive representation. Suppose now that the inverse system $H_I(z)=1/H(z)$ is *causal and stable*. If we assume, without any loss of generality, that $h(0)=1$, then $h_I(n)=Z^{-1}\{H_I(z)\}$ has $h_I(0)=1$. Therefore the input $\omega(n)$ can be obtained by

$$\omega(n)=x(n)+\sum_{k=1}^{\infty} h_I(k)x(n-k) \tag{3.1.5}$$

Solving for $x(n)$, we obtain the following recursive representation for the output signal

$$x(n)=-\sum_{k=1}^{\infty} h_I(k)x(n-k)+\omega(n) \tag{3.1.6}$$

We use the term *recursive* representation to emphasize that the present value of the output is obtained by a linear combination of all past output values, plus the present value of the input. By construction the nonrecursive and recursive representations of system $h(n)$ are equivalent; that is, they produce the same output when they are excited by the same input signal.

Innovations representation. If the system $H(z)$ is minimum-phase, then both $h(n)$ and $h_I(n)$ are causal and stable. Hence, the output signal can be expressed nonrecursively by

$$x(n)=\sum_{k=0}^{\infty}h(k)\omega(n-k)=\sum_{k=-\infty}^{n}h(n-k)\omega(k) \tag{3.1.7}$$

or recursively by (3.1.6).

From (3.1.7) we obtain

$$x(n+1)=\sum_{k=-\infty}^{n}h(n+1-k)\omega(k)+\omega(n+1)$$

or by using (3.1.5)

$$x(n+1)=\underbrace{\sum_{k=-\infty}^{n}h(n+1-k)\sum_{j=-\infty}^{n}h_I(k-j)x(j)}_{\text{past information: linear combination of } x(n),x(n-1),\cdots}+\underbrace{\omega(n+1)}_{\text{new information}} \tag{3.1.8}$$

Careful inspection of (3.1.8) indicates that if the system generating $x(n)$ is minimum-phase, the sample $\omega(n+1)$ brings all the new information (*innovation*) to be carried by the sample $x(n+1)$. All other information can be predicted from the past samples $x(n)$, $x(n-1),\cdots$ of the signal (see Section 5.6). *We stress that this interpretation holds only if* $H(z)$ *is minimum-phase.*

The system $H(z)$ generates the signal $x(n)$ by introducing dependence in the white noise input $\omega(n)$ and is known as the *synthesis* or *coloring* filter. In contrast, the inverse system $H_I(z)$ can be used to recover the input $\omega(n)$ and is known as the *analysis* or *whitening* filter. In this sense the innovations sequence and the output process are completely equivalent. The synthesis and analysis filters are shown in Figure 3.2.

FIGURE 3.2
Synthesis and analysis filters used in innovations representation.

Spectral factorization

Most random processes with a continuous PSD $R_x(e^{j\omega})$ can be generated by exciting a minimum-phase system $H_{\min}(z)$ with white noise. The PSD of the resulting process is given by

$$R_x(\mathrm{e}^{\mathrm{j}\omega})=\sigma_\omega^2|H_{\min}(\mathrm{e}^{\mathrm{j}\omega})|^2 \tag{3.1.9}$$

The process of obtaining $H_{\min}(z)$ from $R_x(e^{j\omega})$ or $r_x(l)$ is known as *spectral factorization.*

If the PSD $R_x(\mathrm{e}^{\mathrm{j}\omega})$ satisfies the Paley-Wiener condition

$$\int_{-\pi}^{\pi}|\ln R_x(\mathrm{e}^{\mathrm{j}\omega})|\,\mathrm{d}\omega<\infty \tag{3.1.10}$$

then the process $x(n)$ is called *regular* and its complex PSD can be factored as follows

$$R_x(z)=\sigma_\omega^2 H_{\min}(z)H_{\min}^*\left(\frac{1}{z^*}\right) \tag{3.1.11}$$

where

$$\sigma_\omega^2=\exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln[R_x(\mathrm{e}^{\mathrm{j}\omega})]\,\mathrm{d}\omega\right\} \tag{3.1.12}$$

is the variance of the white noise input and can be interpreted as the *geometric mean* of $R_x(\mathrm{e}^{\mathrm{j}\omega})$. Consider the inverse Fourier transform of $\ln R_x(\mathrm{e}^{\mathrm{j}\omega})$:

$$c(k)\triangleq\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln[R_x(\mathrm{e}^{\mathrm{j}\omega})]\mathrm{e}^{\mathrm{j}k\omega}\,\mathrm{d}\omega \tag{3.1.13}$$

which is a sequence known as the *cepstrum* of $r_x(l)$. Note that $c(0)=\sigma_\omega^2$. Thus in the cepstral domain, the multiplicative factors $H_{\min}(z)$ and $H_{\min}^*(1/z^*)$ are now *additively separable* due to the natural logarithm of $R_x(\mathrm{e}^{\mathrm{j}\omega})$. Define

$$c_+(k) \triangleq \frac{c(0)}{2} + c(k)u(k-1) \tag{3.1.14}$$

and
$$c_-^*(k) \triangleq \frac{c(0)}{2} + c(k)u(-k-1) \tag{3.1.15}$$

as the positive- and negative-axis projections of $c(k)$, respectively, with $c(0)$ distributed equally between them. Then we obtain

$$h_{\min}(n) = \mathscr{F}^{-1}\{\exp \mathscr{F}[c_+(k)]\} \tag{3.1.16}$$

as the impulse response of the minimum-phase system $H_{\min}(z)$. Similarly,

$$h_{\max}(n) = \mathscr{F}^{-1}\{\exp \mathscr{F}[c_-(k)]\} \tag{3.1.17}$$

is the corresponding maximum-phase system. This completes the spectral factorization procedure for an arbitrary PSD $R_x(e^{j\omega})$, which, in general, is a complicated task. However, it is straightforward if $R_x(z)$ is a rational function.

Spectral flatness measure

The spectral flatness measure (SFM) of a zero-mean process with PSD $R_x(e^{j\omega})$ is defined by (Makhoul 1975)

$$\text{SFM}_x \triangleq \frac{\exp\left\{\dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} \ln[R_x(e^{j\omega})]\,d\omega\right\}}{\dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} R_x(e^{j\omega})\,d\omega} = \frac{\sigma_\omega^2}{\sigma_x^2} \tag{3.1.18}$$

where the second equality follows from (3.1.12). It describes the shape (or more appropriately, flatness) of the PSD by a single number. If $x(n)$ is a white noise process, then $R_x(e^{j\omega}) = \sigma_x^2$ and $\text{SFM}_x = 1$. More specifically, we can show that

$$0 \le \text{SFM}_x \le 1 \tag{3.1.19}$$

Observe that the numerator of (3.1.18) is the geometric mean while the denominator is the arithmetic mean of a real-valued, nonnegative continuous waveform $R_x(e^{j\omega})$. Since $x(n)$ is a regular process satisfying (3.1.10), these means are always positive. Furthermore, their ratio, by definition, is never greater than unity and is equal to unity if the waveform is constant. This, then, proves (3.1.19). A detailed proof is given in Jayant and Noll (1984).

When $x(n)$ is obtained by filtering the zero-mean white noise process $\omega(n)$ through the filter $H(z)$, then the coloring of $R_x(e^{j\omega})$ is due to $H(z)$. In this case, $R_x(e^{j\omega}) = \sigma_\omega^2 |H(e^{j\omega})|^2$ from (3.1.9), and we obtain

$$\text{SFM}_x = \frac{\sigma_\omega^2}{\sigma_x^2} = \frac{\sigma_\omega^2}{\dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} \sigma_\omega^2 |H(e^{j\omega})|^2\,d\omega} = \frac{1}{\dfrac{1}{2\pi}\displaystyle\int_{-\pi}^{\pi} |H(e^{j\omega})|^2\,d\omega} \tag{3.1.20}$$

Thus SFM_x is the inverse of the filter power (or *power transfer factor*) if $h(0)$ is normalized to unity.

3.1.2 Parametric Pole-Zero Signal Models

Parametric models describe a system with a finite number of parameters. The major subject of this chapter is the treatment of parametric models that have rational system functions. To this end, consider a system described by the following linear constant-coefficient difference equation

$$x(n) + \sum_{k=1}^{P} a_k\, x(n-k) = \sum_{k=0}^{Q} d_k\, \omega(n-k) \tag{3.1.21}$$

where $\omega(n)$ and $x(n)$ are the input and output signals, respectively. Taking the z-transform of both sides, we find that the system function is

$$H(z)=\frac{X(z)}{W(z)}=\frac{\sum_{k=0}^{Q}\mathrm{d}_k z^{-k}}{1+\sum_{k=1}^{P}a_k z^{-k}}\triangleq\frac{D(z)}{A(z)} \tag{3.1.22}$$

We can express $H(z)$ in terms of the poles and zeros of the system as follows:

$$H(z)=\mathrm{d}_0\frac{\prod_{k=1}^{Q}(1-z_k z^{-1})}{\prod_{k=1}^{P}(1-p_k z^{-1})} \tag{3.1.23}$$

The system has Q zeros $\{z_k\}$ and P poles $\{p_k\}$ (zeros and poles at $z=0$ are not considered here). The term d_0 is the system gain. For the rest of the book, we assume that the polynomials $D(z)$ and $A(z)$ do not have any common roots, that is, common poles and zeros have already been canceled.

Types of pole-zero models

There are three cases of interest:

- For $P>0$ and $Q>0$, we have a *pole-zero* model, denoted by PZ(P,Q). If the model is assumed to be causal, its output is given by

$$x(n)=-\sum_{k=1}^{P}a_k x(n-k)+\sum_{k=0}^{Q}\mathrm{d}_k w(n-k) \tag{3.1.24}$$

- For $P=0$, we have an *all-zero* model, denoted by AZ(Q). The input-output difference equation is

$$x(n)=\sum_{k=0}^{Q}\mathrm{d}_k w(n-k) \tag{3.1.25}$$

- For $Q=0$, we have an *all-pole* model, denoted by AP(P). The input-output difference equation is

$$x(n)=-\sum_{k=1}^{P}a_k x(n-k)+\mathrm{d}_0 w(n) \tag{3.1.26}$$

If we excite a parametric model with white noise, we obtain a signal whose second-order moments are determined by the parameters of the model. Indeed, from Section 2.2.2, we recall that if $w(n)\sim \mathrm{IID}\{0,\sigma_w^2\}$ with finite variance, then

$$r_x(l)=\sigma_w^2 r_h(l)=\sigma_w^2 h(l)*h^*(-l) \tag{3.1.27}$$

$$R_x(z)=\sigma_w^2 R_h(z)=\sigma_w^2 H(z)H^*\left(\frac{1}{z^*}\right) \tag{3.1.28}$$

$$R_x(\mathrm{e}^{\mathrm{j}\omega})=\sigma_w^2 R_h(\mathrm{e}^{\mathrm{j}\omega})=\sigma_w^2\,|H(\mathrm{e}^{\mathrm{j}\omega})|^2 \tag{3.1.29}$$

Such signal models are of great practical interest and have special names in the statistical literature:

- The AZ(Q) is known as the *moving-average* model, denoted by MA(Q).
- The AP(P) is known as the *autoregressive* model, denoted by AR(P).
- The PZ(P , Q) is known as the *autoregressive moving-average* model, denoted by ARMA (P , Q).

We specify a parametric signal model by normalizing $d_0=1$ and setting the variance of the input to σ_w^2. The defining set of model parameters is given by $\{a_1,a_2,\cdots,a_P,d_1,\cdots,d_Q,\sigma_w^2\}$ (see Figure 3.3). An alternative is to set $\sigma_w^2=1$ and leave d_0 arbitrary. We stress that these models assume the resulting processes are stationary, which is ensured if the corresponding systems are BIBO stable.

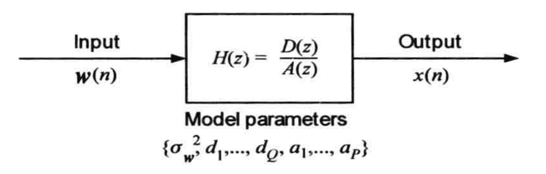


FIGURE 3.3
Block diagram representation of a parametric, rational signal model.

Short-memory behavior

To find the memory behavior of pole-zero models, we investigate the nature of their impulse response. To this end, we recall that for $Q \geq P$, (3.1.23) can be expanded as

$$H(z) = \sum_{j=0}^{Q-P} B_j \, z^{-j} + \sum_{k=1}^{P} \frac{A_k}{1 - p_k \, z^{-1}} \tag{3.1.30}$$

where for simplicity we assume that the model has P distinct poles. The first term in (3.1.30) disappears if $P > Q$. The coefficients B_j can be obtained by long division:

$$A_k = (1 - p_k \, z^{-1}) H(z)\,|_{z=p_k} \tag{3.1.31}$$

If the model is causal, taking the inverse z-transform results in an impulse response that is a linear combination of impulses, real exponentials, and damped sinusoids (produced by the combination of complex exponentials)

$$h(n) = \sum_{j=0}^{Q-P} B_j \delta(n-j) + \sum_{k=1}^{P_1} A_k (p_k)^n u(n) + \sum_{i=1}^{P_2} C_i r_i^n \cos(\omega_i n + \varphi_i) u(n) \tag{3.1.32}$$

where $p_i = r_i e^{\pm j\omega_i}$ and $P = P_1 + 2P_2$. Recall that $u(n)$ and $\delta(n)$ are the unit step and unit impulse functions, respectively. We note that the memory of any all-pole model decays exponentially with time and that the rate of decay is controlled by the pole closest to the unit circle. The contribution of multiple poles at the same location is treated in Problem 3.1.

Careful inspection of (3.1.32) leads to the following conclusions:

1. For AZ(Q) models, the impulse response has finite duration and, therefore, can have any shape.
2. The impulse response of causal AP(P) and PZ(P, Q) models with single poles consists of a linear combination of damped real exponentials (produced by the real poles) and exponentially damped sinusoids (produced by complex conjugate poles). The rate of decay decreases as the poles move closer to the unit circle and is determined by the pole closest to the unit circle.
3. The model is stable if and only if $h(n)$ is absolutely summable, which, due to (3.1.32), is equivalent to $|p_k| < 1$ for all k. In other words, a causal pole-zero model is BIBO stable if and only if all the poles are inside the unit circle.[1]

We conclude that causal, stable PZ(P, Q) models with $P > 0$ have an exponentially fading memory because their impulse response decays exponentially with time. Therefore, the autocorrelation $r_h(l) = h(l) * h^*(-l)$ also decays exponentially (see Example 3.2.2), and pole-zero models have short memory according to the definition given in Section 2.2.3.

Generation of random signals with rational power spectra

Sample realizations of random sequences with rational power spectra can be easily generated by using the difference equation (3.1.24) and a random number generator. In most applications, we use a Gaussian excitation because the generated sequence will also be Gaussian. For non-Gaussian inputs, it is difficult to predict the type of distribution of the output signal. If, on one hand, we specify the frequency response of the model, the coefficients of the difference equation can be obtained by using a digital filter design package. If, on the other hand, the power spectrum or the autocorrelation is given, the coefficients of the model are determined via spectral factorization. If we wish to avoid the transient effects that make some of the initial output samples nonstationary, we should consider the response of the model only after the initial transients have died out.

3.1.3 Mixed Processes and Wold Decomposition

An arbitrary stationary random process can be constructed to possess a continuous PSD $R_x(e^{j\omega})$ and a discrete power spectrum $R_x(k)$. Such processes are called *mixed* processes because the continuous PSD is due to regular processes while the discrete spectrum is due to harmonic (or almost periodic) processes. A further interpretation of mixed processes is that the first part is an *unpredictable* process while the second part is a *predictable* process (in the

[1] Poles on the unit circle are discussed in Section 3.5.

sense that past samples can be used to exactly determine future samples). This interpretation is due to the Wold decomposition theorem.

THEOREM 3.1 (WOLD DECOMPOSITION). A general stationary random process can be written as a sum

$$x(n) = x_r(n) + x_p(n) \tag{3.1.33}$$

where $x_r(n)$ is a regular process possessing a continuous spectrum and $x_p(n)$ is a predictable process possessing a discrete spectrum. Furthermore, $x_r(n)$ is orthogonal to $x_p(n)$; that is,

$$E\{x_r(n_1)x_p^*(n_2)\} = 0 \qquad \text{for all } n_1, n_2 \tag{3.1.34}$$

The proof of this theorem is very involved, but a good approach to it is given in Therrien (1992). Using (3.1.34), the correlation sequence of $x(n)$ in (3.1.33) is given by

$$r_x(l) = r_{x_r}(l) + r_{x_p}(l)$$

from which we obtain the continuous and discrete spectra. As discussed above, the regular process has an innovations representation $\omega(n)$ that is uncorrelated but *not* independent. For example, $\omega(n)$ can be the output of an all-pass filter driven by an IID sequence.

3.2 All-Pole Models

We start our discussion of linear signal models with all-pole models because they are the easiest to analyze and the most often used in practical applications. We assume an all-pole model of the form

$$H(z) = \frac{d_0}{A(z)} = \frac{d_0}{1 + \sum_{k=1}^{P} a_k z^{-k}} = \frac{d_0}{\prod_{k=1}^{P}(1 - p_k z^{-1})} \tag{3.2.1}$$

where d_0 is the system gain and P is the order of the model. The all-pole model can be implemented using either a direct or a lattice structure. The conversion between the two sets of parameters can be done by using the step-up and step-down recursions.

3.2.1 Model Properties

In this section, we derive analytic expressions for various properties of the all-pole model, namely, the impulse response, the autocorrelation, and the spectrum. We determine the system-related properties $r_h(l)$ and $R_h(e^{j\omega})$ because the results can be readily applied to obtain the signal model properties for inputs with both continuous and discrete spectra.

Impulse response. The impulse response $h(n)$ can be specified by first rewriting (3.2.1) as

$$H(z) + \sum_{k=1}^{P} a_k H(z) z^{-k} = d_0$$

and then taking the inverse z-transform to obtain

$$h(n) + \sum_{k=1}^{P} a_k h(n-k) = d_0 \delta(n) \tag{3.2.2}$$

If the system is causal, then

$$h(n) = -\sum_{k=1}^{P} a_k h(n-k) + d_0 \delta(n) \tag{3.2.3}$$

If $H(z)$ has all its poles inside the unit circle, then $h(n)$ is a causal, stable sequence and the system is minimum-phase. From (3.2.3) we have

$$h(0) = d_0 \tag{3.2.4}$$

$$h(n) = -\sum_{k=1}^{P} a_k h(n-k) \qquad n > 0 \tag{3.2.5}$$

and owing to causality we have

$$h(n) = 0 \qquad n < 0 \tag{3.2.6}$$

Thus, except for the value at $n = 0$, $h(n)$ can be obtained recursively as a linearly weighted summation of its previous values $h(n-1), \cdots, h(n-P)$. One can say that $h(n)$ can be *predicted* (with zero error for $n \neq 0$) from the past P values. Thus, the coefficients $\{a_k\}$ are often referred to as *predictor coefficients*. Note that there is a close relationship between all-pole models and linear prediction that will be discussed in Section 3.2.2.

From (3.2.4) and (3.2.5), we can also write the inverse relation

$$a_n = -\frac{h(n)}{h(0)} - \sum_{k=1}^{n-1} a_k \frac{h(n-k)}{h(0)} \qquad n > 0 \tag{3.2.7}$$

with $a_0 = 1$. From (3.2.7) and (3.2.4), we conclude that if we are given the first $P+1$ values of the impulse response $h(n)$, $0 \le n \le P$, then the parameters of the all-pole filter are completely specified.

Finally, we note that a causal $H(z)$ can be written as a one-sided, infinite polynomial $H(z) = \sum_{n=0}^{\infty} h(n) z^{-n}$. This representation of $H(z)$ implies that any finite-order, all-pole model can be represented equivalently by an infinite number of zeros. In general, a single pole can be represented by an infinite number of zeros, and conversely a single zero can be represented by an infinite number of poles. If the poles are inside the unit circle, so are the corresponding zeros, and vice versa.

EXAMPLE 3.2.1. A single pole at $z = a$ can be represented by

$$H(z) = \frac{1}{1 - az^{-1}} = \sum_{n=0}^{\infty} a^n z^{-n} \qquad |a| < 1 \tag{3.2.8}$$

The question is, where are the infinite number of zeros located? To find the answer, let us consider the finite polynomial

$$H_N(z) = \sum_{n=0}^{N} a^n z^{-n} \tag{3.2.9}$$

where we have truncated $H(z)$ at $n = N$. Thus $H_N(z)$ is a geometric series that can be written in closed form as

$$H_N(z) = \frac{1 - a^{N+1} z^{-(N+1)}}{1 - az^{-1}} \tag{3.2.10}$$

And $H_N(z)$ has a single pole at $z = a$ and $N+1$ zeros at

$$z_i = a e^{j2\pi i/(N+1)} \qquad i = 0, 1, \cdots, N \tag{3.2.11}$$

The $N+1$ zeros are equally distributed on the circle $|z| = a$ with one of the zeros (for $i = 0$) located at $z = a$. But the zero at $z = a$ cancels the pole at the same location. Therefore, $H_N(z)$ has the remaining N zeros:

$$z_i = a e^{j2\pi i(N+1)} \qquad i = 1, 2, \cdots, N \tag{3.2.12}$$

The transfer function $H(z)$ of the single-pole model is obtained from $H_N(z)$ by letting N go to infinity. In the limit, $H_\infty(z)$ has an infinite number of zeros equally distributed on the circle $|z| = a$; the zeros are everywhere on that circle except at the point $z = a$. Similarly, the denominator from (3.2.8), a polynomial with a single zero at $z = a$, can be written as

$$A(z) = 1 - az^{-1} = \frac{1}{H(z)} = \frac{1}{1 + \sum_{n=0}^{\infty} a^n z^{-n}} \qquad |a| < 1 \tag{3.2.13}$$

that is, a single zero can also be represented by an infinite number of poles. In this case, the poles are equally distributed on a circle that passes through the location of the zero; the poles are everywhere on the circle except at the actual location of the zero.

Autocorrelation. The impulse response $h(n)$ of an all-pole model has infinite duration so that its autocorrelation involves an infinite summation, which is not practical to write in closed form except for low-order models. However, the autocorrelation function obeys a recursive relation that relates the autocorrelation values to the model parameters. Multiplying (3.2.2) by $h^*(n-l)$ and summing over all n, we have

$$\sum_{n=-\infty}^{\infty} \sum_{k=0}^{P} a_k h(n-k) h^*(n-l) = d_0 \sum_{n=-\infty}^{\infty} h^*(n-l) \delta(n) \tag{3.2.14}$$

where $a_0 = 1$. Interchanging the order of summations in the left-hand side, we obtain

$$\sum_{k=0}^{P} a_k r_h(l-k) = d_0 h^*(-l) \qquad -\infty < l < \infty \tag{3.2.15}$$

where $r_h(l)$ is the autocorrelation of $h(n)$. Equation (3.2.15) is true for all l, but because $h(l)=0$ for $l<0$, $h(-l)=0$ for $l>0$, and we have

$$\sum_{k=0}^{P} a_k r_h(l-k) = 0 \qquad l>0 \tag{3.2.16}$$

From (3.2.4) and (3.2.15), we also have for $l=0$,

$$\sum_{k=0}^{P} a_k r_h(k) = |d_0|^2 \tag{3.2.17}$$

where we used the fact that $r_h^*(-l) = r_h(l)$. Equation (3.2.16) can be rewritten as

$$r_h(l) = -\sum_{k=1}^{P} a_k r_h(l-k) \qquad l>0 \tag{3.2.18}$$

which is a recursive relation for $r_h(l)$ in terms of past values of the autocorrelation and $\{a_k\}$. Relation (3.2.18) for $r_h(l)$ is similar to relation (3.2.5) for $h(n)$, but with one important difference: (3.2.5) for $h(n)$ is true for all $n \neq 0$ while (3.2.18) for $r_h(l)$ is true only if $l>0$; for $l<0$, $r_h(l)$ obeys (3.2.15).

If we define the *normalized* autocorrelation coefficients as

$$\rho_h(l) = \frac{r_h(l)}{r_h(0)} \tag{3.2.19}$$

then we can divide (3.2.17) by $r_h(0)$ and deduce the following relation for $r_h(0)$

$$r_h(0) = \frac{|d_0|^2}{1+\sum_{k=1}^{P} a_k \rho_h(k)} \tag{3.2.20}$$

which is the energy of the output of the all-pole filter when excited by a single impulse.

Autocorrelation in terms of poles. The complex spectrum of the AP(P) model is

$$R_h(z) = H(z)H\left(\frac{1}{z^*}\right) = |d_0|^2 \prod_{k=1}^{P} \frac{1}{(1-p_k z^{-1})(1-p_k z^*)} \tag{3.2.21}$$

Therefore, the autocorrelation sequence can be expressed in terms of the poles by taking the inverse z-transform of $R_h(z)$, that is, $r_h(l) = Z^{-1}\{R_h(z)\}$. The poles p_k of the minimum-phase model $H(z)$ contribute causal terms in the partial fraction expansion, whereas the poles $1/p_k$ of the nonminimum-phase model $H(1/z^*)$ contribute noncausal terms. This is best illustrated with the following example.

EXAMPLE 3.2.2. Consider the following minimum-phase AP(1) model

$$H(z) = \frac{1}{1+az^{-1}} \qquad -1<a<1 \tag{3.2.22}$$

Owing to causality, the ROC of $H(z)$ is $|z|>|a|$. The z-transform

$$H(z^{-1}) = \frac{1}{1+az} \qquad -1<a<1 \tag{3.2.23}$$

corresponds to the noncausal sequence $h(-n) = (-a)^{-n}u(-n)$, and its ROC is $|z|<1/|a|$. Hence,

$$R_h(z) = H(z)H(z^{-1}) = \frac{1}{(1+az^{-1})(1+az)} \tag{3.2.24}$$

which corresponds to a two-sided sequence because its ROC, $|a|<|z|<1/|a|$, is a ring in the z-plane. Using partial fraction expansion, we obtain

$$R_h(z) = \frac{-a}{1-a^2}\frac{z^{-1}}{1+az^{-1}} + \frac{1}{1-a^2}\frac{1}{1+az} \tag{3.2.25}$$

The pole $p=-a$ corresponds to the causal sequence $[1/(1-a^2)](-a)^l u(l-1)$, and the pole $p=-1/a$ to the noncausal sequence $[1/(1-a^2)](-a)^{-l}u(-l)$. Combining the two terms, we obtain

$$r_h(l) = \frac{1}{1-a^2}(-a)^{|l|} \qquad -\infty < l < \infty \tag{3.2.26}$$

or

$$\rho_h(l) = (-a)^{|l|} \qquad -\infty < l < \infty \tag{3.2.27}$$

Note that complex conjugate poles will contribute two-sided damped sinusoidal terms like the ones described in Section 3.1.2 for the AP(2) model.

Impulse train excitations. The response of an AP(P) model to a periodic impulse train with period L is periodic with the same period and is given by

$$\begin{aligned} \tilde{h}(n) + \sum_{k=1}^{P} a_k \tilde{h}(n-k) &= d_0 \sum_{m=-\infty}^{\infty} \delta(n+Lm) \\ &= \begin{cases} d_0 & n+Lm=0 \\ 0 & n+Lm \neq 0 \end{cases} \end{aligned} \tag{3.2.28}$$

which shows that the prediction error is zero for samples inside the period and d_0 at the beginning of each period. If we multiply both sides of (3.2.28) by $\tilde{h}(n-l)$ and sum over a period $0 \le n \le L-1$, we obtain

$$\tilde{r}_h(l) + \sum_{k=1}^{P} a_k \tilde{r}_h(l-k) = \frac{d_0}{L} \tilde{h}^*(-l) \qquad \text{all } l \tag{3.2.29}$$

where $\tilde{r}_h(l)$ is the periodic autocorrelation of $\tilde{h}(n)$. Since, in contrast to $h(n)$ in (3.2.15), $\tilde{h}(n)$ is not necessarily zero for $n<0$, the periodic autocorrelation $\tilde{r}_h(l)$ will not in general obey the linear prediction equation anywhere. Similar results can be obtained for harmonic process excitations.

Model parameters in terms of autocorrelation. Equations (3.2.15) for $l = 0, 1, \cdots, P$ comprise $P+1$ equations that relate the $P+1$ parameters of $H(z)$, namely, d_0 and $\{a_k, 1 \le k \le P\}$, to the first $P+1$ autocorrelation coefficients $r_h(0), r_h(1), \cdots, r_h(P)$. These $P+1$ equations can be written in matrix form as

$$\begin{bmatrix} r_h(0) & r_h(1) & \cdots & r_h(P) \\ r_h^*(1) & r_h(0) & \cdots & r_h(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_h^*(P) & r_h^*(P-1) & \cdots & r_h(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} |d_0|^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.2.30}$$

If we are given the first $P+1$ autocorrelations, (3.2.30) comprises a system of $P+1$ linear equations, with a Hermitian Toeplitz matrix that can be solved for d_0 and $\{a_k\}$.

Because of the special structure in (3.2.30), the model parameters are found from the autocorrelations by using the last set of P equations in (3.2.17), followed by the computation of d_0 from the first equation, which is the same as (3.2.17). From (3.2.30), we can write in matrix notation

$$\mathbf{R}_h \mathbf{a} = -\mathbf{r}_h \tag{3.2.31}$$

where $\mathbf{R}_h$ is the autocorrelation matrix, $\mathbf{a}$ is the vector of the model parameters, and $\mathbf{r}_h$ is the vector of autocorrelations. Since $r_x(l) = \sigma_\omega^2 r_h(l)$, we can also express the model parameters in terms of the autocorrelation $r_x(l)$ of the output process $x(n)$ as follows:

$$\mathbf{R}_x \mathbf{a} = -\mathbf{r}_x \tag{3.2.32}$$

These equations are known as the *Yule-Walker equations* in the statistics literature. In the sequel, we drop the subscript from the autocorrelation sequence or matrix whenever the analysis holds for both the impulse response and the model output.

Because of the Toeplitz structure and the nature of the right-hand side, the linear systems (3.2.31) and (3.2.32) can be solved recursively by using the algorithm of Levinson-Durbin. After $\mathbf{a}$ is solved for, the system gain d_0 can be computed from (3.2.17).

Therefore, given $r(0)$, $r(1), \cdots, r(P)$, we can completely specify the parameters of the all-pole model by

solving a set of linear equations. Below, we will see that the converse is also true: Given the model parameters, we can find the first $P+1$ autocorrelations by solving a set of linear equations. *This elegant solution of the spectral factorization problem is unique to all-pole models.* In the case in which the model contains zeros ($Q \neq 0$), the spectral factorization problem requires the solution of a nonlinear system of equations.

Autocorrelation in terms of model parameters. If we normalize the autocorrelations in (3.2.31) by dividing throughout by $r(0)$, we obtain the following system of equations

$$\boldsymbol{Pa} = -\boldsymbol{\rho} \tag{3.2.33}$$

where $\boldsymbol{P}$ is the normalized autocorrelation matrix and

$$\boldsymbol{\rho} = [\rho(1) \ \ \rho(2) \ \ \cdots \ \ \rho(P)]^H \tag{3.2.34}$$

is the vector of normalized autocorrelations. This set of P equations relates the P model coefficients with the first P (normalized) autocorrelation values. If the poles of the all-pole filter are strictly inside the unit circle, the mapping between the P-dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{\rho}$ is *unique*. If, in fact, we are given the vector $\boldsymbol{a}$, then the normalized autocorrelation vector $\boldsymbol{\rho}$ can be computed from $\boldsymbol{a}$ by using the set of equations that can be deduced from (3.2.33)

$$\boldsymbol{A\rho} = -\boldsymbol{a} \tag{3.2.35}$$

where $<\boldsymbol{A}>_{ij} = a_{i-j} + a_{i+j}$, assuming $a_m = 0$ for $m < 0$ and $m > P$ (see Problem 3.6).

Given the set of coefficients in $\boldsymbol{a}$, $\boldsymbol{\rho}$ can be obtained by solving (3.2.25). We will see that, under the assumption of a stable $H(z)$, a solution always exists. Furthermore, there exists a simple, recursive solution that is efficient (see Section 6.5). If, in addition to $\boldsymbol{a}$, we are given d_0, we can evaluate $r(0)$ with (3.2.20) from $\boldsymbol{\rho}$ computed by (3.2.25). Autocorrelation values $r(l)$ for lags $l > P$ are found by using the recursion in (3.2.18) with $r(0), r(1), \cdots, r(P)$.

EXAMPLE 3.2.3. For the AP(3) model with real coefficients we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = -\begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix} \tag{3.2.36}$$

$$d_0^2 = r(0) + a_1 r(1) + a_2 r(2) + a_3 r(3) \tag{3.2.37}$$

Therefore, given $r(0)$, $r(1)$, $r(2)$, $r(3)$, we can find the parameters of the all-pole model by solving (3.2.36) and then substituting into (3.2.37).

Suppose now that instead we are given the model parameters d_0, a_1, a_2, a_3. If we divide both sides of (3.2.36) by $r(0)$ and solve for the normalized autocorrelations $\rho(1), \rho(2)$, and $\rho(3)$, we obtain

$$\begin{bmatrix} 1+a_2 & a_3 & 0 \\ a_1 + a_3 & 1 & 0 \\ a_2 & a_1 & 1 \end{bmatrix} \begin{bmatrix} \rho(1) \\ \rho(2) \\ \rho(3) \end{bmatrix} = -\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.2.38}$$

The value of $r(0)$ is obtained from

$$r(0) = \frac{d_0^2}{1 + a_1\rho(1) + a_2\rho(2) + a_3\rho(3)} \tag{3.2.39}$$

If $r(0) = 2$, $r(1) = 1.6$, $r(2) = 1.2$, and $r(3) = 1$, the Toeplitz matrix in (3.2.36) is positive definite because it has positive eigenvalues. Solving the linear system gives $a_1 = -0.9063$, $a_2 = 0.2500$, and $a_3 = -0.1563$. Substituting these values in (3.2.37), we obtain $d_0 = 0.8329$. Using the last two relations, we can recover the autocorrelation from the model parameters.

Correlation matching. All-pole models have the unique distinction that the model parameters are completely specified by the first $P+1$ autocorrelation coefficients via a set of linear equations. We can write

$$\begin{bmatrix} d_0 \\ \boldsymbol{a} \end{bmatrix} \leftrightarrow \begin{bmatrix} r(0) \\ \boldsymbol{\rho} \end{bmatrix} \tag{3.2.40}$$

that is, the mapping of the model parameters $\{d_0, \ a_1, \ a_2, \ \cdots, \ a_P\}$ to the autocorrelation coefficients specified by the vector $\{r(0), \ \rho(1), \ \cdots, \ \rho(P)\}$ is reversible and unique. This statement implies that given any set of autocorrelation values $r(0)$, $r(1)$, $\cdots$, $r(P)$, we can always find an all-pole model whose first $P+1$

autocorrelation coefficients are equal to the given autocorrelations. This *correlation matching* of all-pole models is quite remarkable. This property is not shared by all-zero models and is true for pole-zero models only under certain conditions, as we will see in Section 3.4.

Spectrum. The z-transform of the autocorrelation $r(l)$ of $H(z)$ is given by

$$R(z)=H(z)H\left(\frac{1}{z^*}\right) \tag{3.2.41}$$

The spectrum is then equal to

$$R(e^{j\omega})=|H(e^{j\omega})|^2=\frac{|d_0|^2}{|A(e^{j\omega})|^2} \tag{3.2.42}$$

The right-hand side of (3.2.42) suggests a method for computing the spectrum: First compute $A(e^{j\omega})$ by taking the Fourier transform of the sequence $\{1,\ a_1,\ \cdots,\ a_P\}$, then take the squared of the magnitude and divide $|d_0|^2$ by the result. The fast Fourier transform (FFT) can be used to this end by appending the sequence $\{1,\ a_1,\ \cdots,\ a_P\}$ with as many zeros as needed to compute the desired number of frequency points.

Partial autocorrelation and lattice structures. We have seen that an AP(P) model is completely described by the first $P+1$ values of its autocorrelation. However, we cannot determine the order of the model by using the autocorrelation sequence because it has infinite duration. Suppose that we start fitting models of increasing order m, using the autocorrelation sequence of an AP(P) model and the Yule-Walker equations

$$\begin{bmatrix} 1 & \rho(1) & \cdots & \rho(m-1) \\ \rho^*(1) & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \rho(1) \\ \rho^*(m-1) & \cdots & \rho^*(1) & 1 \end{bmatrix} \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ \vdots \\ a_m^{(m)} \end{bmatrix} = -\begin{bmatrix} \rho^*(1) \\ \rho^*(2) \\ \vdots \\ \rho^*(m) \end{bmatrix} \tag{3.2.43}$$

Recall the relationship of the coefficients between that of a direct-form filter and that of an lattice filter.

$$a_m^{(m)}=k_m \tag{3.2.44}$$

that is, the PACS is identical to the lattice parameters. A statistical definition and interpretation of the PACS are also given in Chapter 7. The PACS can be defined for any valid (i.e., positive definite) autocorrelation sequence and can be efficiently computed by using the algorithms of Levinson-Durbin and Schur (see Chapter 6).

Furthermore, it has been shown (Burg 1975) that

$$r(0)\prod_{m=1}^{P}\frac{1-|k_m|}{1+|k_m|}\le R(e^{j\omega})\le r(0)\prod_{m=1}^{P}\frac{1+|k_m|}{1-|k_m|} \tag{3.2.45}$$

which indicates that the spectral dynamic range increases if some lattice parameter moves close to 1 or equivalently some pole moves close to the unit circle.

Equivalent model representations. From the previous discussions we conclude that a minimum-phase AP(P) model can be uniquely described by any one of the following representations:

1. Direct structure: $\{d_0, a_1, a_2, \cdots, a_P\}$
2. Lattice structure: $\{d_0, k_1, k_2, \cdots, k_P\}$
3. Autocorrelation: $\{r(0), r(1), \cdots, r(P)\}$

where we assume, without loss of generality, that $d_0>0$. Note that the minimum-phase property requires that all poles be inside the unit circle or all $|k_m|<1$ or that $\mathbf{R}_{P+1}$ be positive definite. The transformation from any of the above representations to any other can be done by using the algorithms developed in Section 6.5.

Minimum-phase conditions. As we will show in Section 6.5, if the Toeplitz matrix $\mathbf{R}_h$ (or equivalently $\mathbf{R}_x$) is positive definite, then $|k_m|<1$ for all $m=1,\ 2,\ \cdots,\ P$. Therefore, the AP(P) model obtained by solving the Yule-Walker equations is minimum-phase. Therefore, the Yule-Walker equations provide a simple and elegant solution to the spectral factorization problem for all-pole models.

EXAMPLE 3.2.4. The poles of the model obtained in Example 3.2.3 are 0.8316, 0.0373+0.4319i, and $0.0373-0.4319i$. We see that the poles are inside the unit circle and that the autocorrelation sequence is positive definite. If we set $r_h(2)=-1.2$, the autocorrelation becomes negative definite and the obtained model $\boldsymbol{a}=[1\ \ -1.222\ \ 1.1575]^T$, $d_0=2.2271$, is

nonminimum-phase.

Pole locations. The poles of $H(z)$ are the zeros $\{p_k\}$ of the polynomial $A(z)$. If the coefficients of $A(z)$ are assumed to be real, the poles are either real or come in complex conjugate pairs. In order for $H(z)$ to be minimum-phase, all poles must be inside the unit circle, that is, $|p_k|<1$. The model parameters a_k can be written as sums of products of the poles p_k. In particular, it is easy to see that

$$a_1 = -\sum_{k=1}^{P} p_k \tag{3.2.46}$$

$$a_P = \prod_{k=1}^{P}(-p_k) \tag{3.2.47}$$

Thus, the first coefficient a_1 is the negative of the sum of the poles, and the last coefficient a_P is the product of the negative of the individual poles. Since $|p_k|<1$, we must have $|a_P|<1$ for a minimum-phase polynomial for which $a_0=1$. However, note that the reverse is not necessarily true: $|a_P|<1$ does not guarantee minimum phase. The roots p_k can be computed by using any number of standard root-finding routines.

3.2.2 All-Pole Modeling and Linear Prediction

Consider the AP(P) model

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + w(n) \tag{3.2.48}$$

Now recall from Chapter 1 that the M th-order linear predictor of $x(n)$ and the corresponding prediction error $e(n)$ are

$$\hat{x}(n) = -\sum_{k=1}^{M} a_k^0 x(n-k) \tag{3.2.49}$$

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^{M} a_k^0 x(n-k) \tag{3.2.50}$$

or

$$x(n) = \sum_{k=1}^{M} a_k^0 x(n-k) + e(n) \tag{3.2.51}$$

Notice that if the order of the linear predictor equals the order of the all-pole model (*M*=*P*) and if $a_k^0 = a_k$, then the prediction error is equal to the excitation of the all-pole model, that is, $e(n)=w(n)$. Since all-pole modeling and FIR linear prediction are closely related, many properties and algorithms developed for one of them can be applied to the other. Linear prediction is extensively studied in Chapters 5 and 6.

3.2.3 Autoregressive Models

Causal all-pole models excited by white noise play a major role in practical applications and are known as *autoregressive (AR)* models. An AR(*P*) model is defined by the difference equation

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + w(n) \tag{3.2.52}$$

where $\{w(n)\} \sim \mathrm{WN}(0,\sigma_w^2)$. An AR(*P*) model is valid only if the corresponding AP(*P*) system is stable. In this case, the output $x(n)$ is a stationary sequence with a mean value of zero. Postmultiplying (3.2.52) by $x^*(n-l)$ and taking the expectation, we obtain the following recursive relation for the autocorrelation:

$$r_x(l) = -\sum_{k=1}^{P} a_k r_x(l-k) + E\{w(n)x^*(n-l)\} \tag{3.2.53}$$

Similarly, using (3.1.1), we can show that $E\{\omega(n)x^*(n-l)\} = \sigma_w^2 h^*(-l)$. Thus, we have

$$r_x(l) = -\sum_{k=1}^{P} a_k r_x(l-k) + \sigma_w^2 h^*(-l) \qquad \text{for all } l \tag{3.2.54}$$

The variance of the output signal is

$$\sigma_x^2 = r_x(0) = -\sum_{k=1}^{P} a_k r_x(k) + \sigma_w^2$$

or

$$\sigma_x^2 = \frac{\sigma_w^2}{1 + \sum_{k=1}^{P} a_k \rho_x(k)} \tag{3.2.55}$$

If we substitute l=0, 1, ..., P in (3.2.55) and recall that $h(n) = 0$ for $n < 0$, we obtain the following set of Yule-Walker equations:

$$\begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(P) \\ r_x^*(1) & r_x(0) & \cdots & r_x(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x^*(P) & r_x^*(P-1) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.2.56}$$

Careful inspection of the above equations reveals their similarity to the corresponding relationships developed previously for the AP(P) model. This should be no surprise since the power spectrum of the white noise is flat. However, there is one important difference we should clarify: AP(P) models were specified with a gain d_0 and the parameters $\{a_1, a_2, \cdots, a_P\}$, but for AR($P$) models we set the gain $d_0 = 1$ and define the model by the variance of the white excitation σ_w^2 and the parameters $\{a_1, a_2, \cdots, a_P\}$. In other words, we incorporate the gain of the model into the power of the input signal. Thus, the power spectrum of the output is $R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2$. Similar arguments apply to all parametric models driven by white noise. We just rederived some of the relationships to clarify these issues and to provide additional insight into the subject.

3.2.4 Lower-Order Models

In this section, we derive the properties of lower-order all-pole models, namely, first- and second-order models, with real coefficients.

First-order all-pole model: AP(1)

An AP(1) model has a transfer function

$$H(z) = \frac{d_0}{1 + az^{-1}} \tag{3.2.57}$$

with a single pole at $z = -a$ on the real axis. It is clear that $H(z)$ is minimum-phase if

$$-1 < a < 1 \tag{3.2.58}$$

From (3.2.18) with $P = 1$ and $l = 1$, we have

$$a_1 = -\frac{r(1)}{r(0)} = -\rho(1) \tag{3.2.59}$$

Similarly, from (3.2.44) with $m = 1$,

$$a_1^{(1)} = a = -\rho(1) = k_1 \tag{3.2.60}$$

Since from (3.2.4), $h(0) = d_0$, and from (3.2.5) $h(n) = -a_1 h(n-1)$ for $n > 0$, the impulse response of a single-pole filter is given by

$$h(n) = d_0(-a)^n u(n) \tag{3.2.61}$$

The same result can, of course, be obtained by taking the inverse z-transform of $H(z)$.

The autocorrelation is found in a similar fashion. From (3.2.18) and by using the fact that the autocorrelation is an even function,

$$r(l) = r(0)(-a)^{|l|} \qquad \text{for all } l \tag{3.2.62}$$

and from (3.2.20)

$$r(0) = \frac{d_0^2}{1-a^2} = \frac{d_0^2}{1-k_1^2} \tag{3.2.63}$$

Therefore, if the energy $r(0)$ in the impulse response is set to unity, then the gain must be set to

$$d_0 = \sqrt{1-k_1^2} \qquad r(0) = 1 \tag{3.2.64}$$

The z-transform of the autocorrelation is then

$$R(z) = \frac{d_0^2}{(1+az^{-1})(1+az)} = r(0)\sum_{l=-\infty}^{\infty}(-a)^{|l|} z^{-l} \tag{3.2.65}$$

and the spectrum is

$$R(e^{j\omega}) = |H(e^{j\omega})|^2 = \frac{d_0^2}{|1+ae^{-j\omega}|^2} = \frac{d_0^2}{1+2a\cos\omega + a^2} \tag{3.2.66}$$

Figures 3.4 and 3.5 show a typical realization of the output, the impulse response, autocorrelation, and spectrum of two AP(1) models. The sample process realizations were obtained by driving the model with white Gaussian noise of zero mean and unit variance. When the positive pole ($p = -a = 0.8$) is close to the unit circle, successive samples of the output process are similar, as dictated by the slowly decaying autocorrelation and the corresponding low-pass spectrum. In contrast, a negative pole close to the unit circle results in a rapidly oscillating sequence. This is clearly reflected in the alternating sign of the autocorrelation sequence and the associated high-pass spectrum.

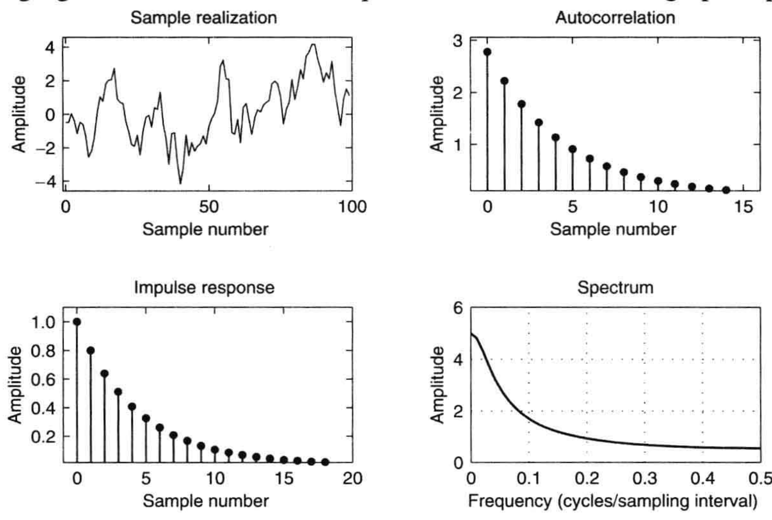


FIGURE 3.4

Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(1) model with $a = -0.8$.

Note that a positive real pole is a type of low-pass filter, while a negative real pole has the spectral characteristics of a high-pass filter. (This situation in the digital domain contrasts with that in the corresponding analog domain where a real-axis pole can only have low-pass characteristics.) The discrete-time negative real pole can be thought of as one-half of two conjugate poles at half the sampling frequency. Notice that both spectra are even and have zero slope at $\omega = 0$ and $\omega = \pi$. These propositions are true of the spectra of all parametric models (i.e., pole-zero models) with real coefficients (see Problem 3.13).

Consider now the real-valued AR(1) process $x(n)$ generated by

$$x(n) = -ax(n-1) + w(n) \tag{3.2.67}$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. Using the formula $R_x(z) = \sigma_\omega^2 H(z)H^*(1/z^*)$ and previous results, we can see that the autocorrelation and the PSD of $x(n)$ are given by

$$r_x(l) = \frac{\sigma_w^2}{1-a^2}(-a)^{|l|}$$

and

$$R_x(e^{j\omega}) = \sigma_w^2 \frac{1-a^2}{1+a^2+2a\cos\omega}$$

respectively. Since $\sigma_x^2 = r_x(0) = \sigma_\omega^2/(1-a^2)$, the SFM of $x(n)$ is [see (Section 3.1.18)]

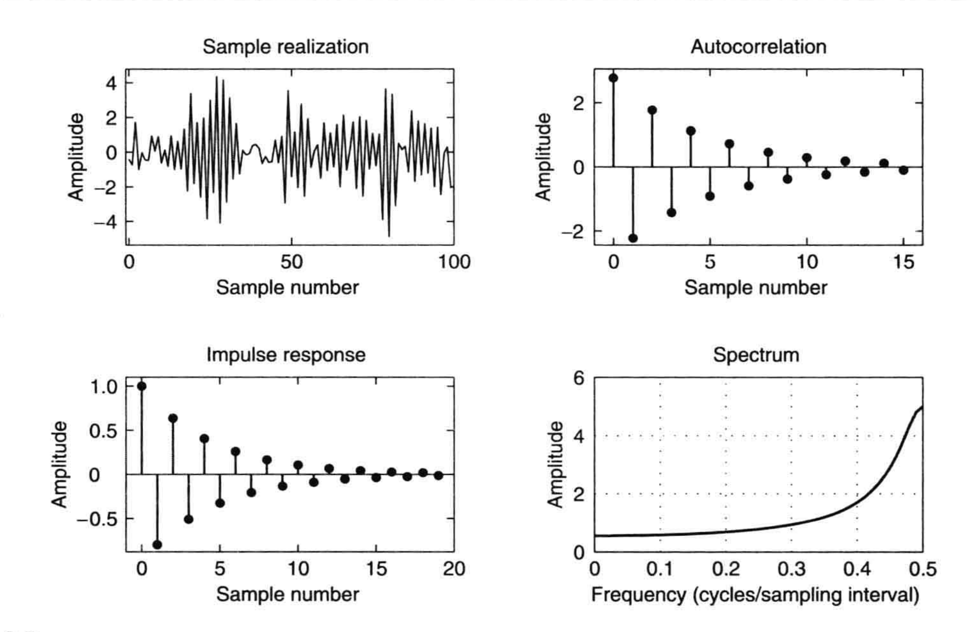


FIGURE 3.5
Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(1) model with $a = 0.8$.

$$\text{SFM}_x = \frac{\sigma_\omega^2}{\sigma_x^2} = 1 - a^2 \tag{3.2.68}$$

Clearly, if $a = 0$, then from (3.2.67), $x(n)$ is a white noise process and from (3.2.68), $\text{SFM}_x = 1$. If $a \to 1$, then $\text{SFM}_x \to 0$; and in the limit when $a = 1$, the process becomes a random walk process, which is a nonstationary process with linearly increasing variance $E\{x^2(n)\} = n\sigma_\omega^2$. The correlation matrix is Toeplitz, and it is a rare exception in which eigenvalues and eigenvectors can be described by analytical expressions (Jayant and Noll 1984).

Second-order all-pole model: AP(2)

The system function of an AP(2) model is given by

$$H(z) = \frac{d_0}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{d_0}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})} \tag{3.2.69}$$

From (3.2.46) and (3.2.47), we have

$$\begin{aligned} a_1 &= -(p_1 + p_2) \\ a_2 &= p_1 p_2 \end{aligned} \tag{3.2.70}$$

Recall that $H(z)$ is minimum-phase if the two poles p_1 and p_2 are inside the unit circle. Under these conditions, a_1 and a_2 lie in a triangular region defined by

$$\begin{aligned} -1 < a_2 < 1 \\ a_2 - a_1 > -1 \\ a_2 + a_1 > -1 \end{aligned} \tag{3.2.71}$$

and shown in Figure 3.6. The first condition follows from (3.2.70) since $|p_1| < 1$ and $|p_2| < 1$. The last two conditions can be derived by assuming real roots and setting the larger root to less than 1 and the smaller root to greater than -1. By adding the last two conditions, we obtain the redundant condition $a_2 > -1$.

Complex roots occur in the region

$$\frac{a_1^2}{4} < a_2 \le 1 \qquad \text{complex poles} \tag{3.2.72}$$

with $a_2 = 1$ resulting in both roots being on the unit circle. Note that, in order to have complex poles, a_2 cannot be negative. If the complex poles are written in polar form

$$p_i = re^{\pm j\theta} \qquad 0 \le r \le 1 \tag{3.2.73}$$

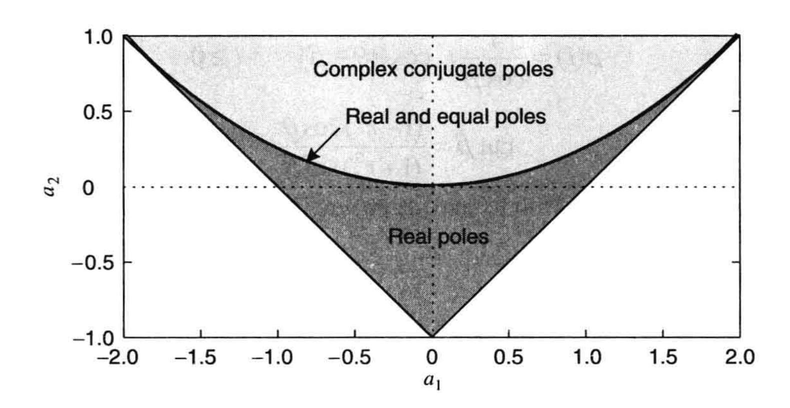


FIGURE 3.6
Minimum-phase region (triangle) for the AP(2) model in the (a_1, a_2) parameter space.

then
$$a_1 = -2r\cos\theta \qquad a_2 = r^2 \tag{3.2.74}$$

and
$$H(z) = \frac{d_0}{1-(2r\cos\theta)z^{-1} + r^2 z^{-2}} \qquad \text{complex poles} \tag{3.2.75}$$

Here, r is the radius (magnitude) of the poles, and θ is the angle or normalized frequency of the poles.

Impulse response. The impulse response of an AP(2) model can be written in terms of its two poles by evaluating the inverse z-transform of (3.2.69). The result is

$$h(n) = \frac{d_0}{p_1 - p_2}(p_1^{n+1} - p_2^{n+1})u(n) \tag{3.2.76}$$

for $p_1 \neq p_2$. Otherwise, for $p_1 = p_2 = p$,

$$h(n) = d_0(n+1)p^n u(n) \tag{3.2.77}$$

In the special case of a complex conjugate pair of poles $p_1 = re^{j\theta}$ and $p_2 = re^{-j\theta}$, Equation (3.2.76) reduces to

$$h(n) = d_0 r^n \frac{\sin[(n+1)\theta]}{\sin\theta}u(n) \qquad \text{complex poles} \tag{3.2.78}$$

Since $0 < r < 1$, $h(n)$ is a damped sinusoid of frequency θ.

Autocorrelation. The autocorrelation can also be written in terms of the two poles as

$$r(l) = \frac{d_0^2}{(p_1 - p_2)(1 - p_1 p_2)}\left(\frac{p_1^{l+1}}{1-p_1^2} - \frac{p_2^{l+1}}{1-p_2^2}\right) \qquad l \geq 0 \tag{3.2.79}$$

from which we can deduce the energy

$$r(0) = \frac{d_0^2(1+p_1 p_2)}{(1-p_1 p_2)(1-p_1^2)(1-p_2^2)} \tag{3.2.80}$$

For the special case of a complex conjugate pole pair, (3.2.79) can be rewritten as

$$r(l) = \frac{d_0^2 r^l \{\sin[(l+1)\theta] - r^2 \sin[(l-1)\theta]\}}{[(1-r^2)\sin\theta](1-2r^2\cos 2\theta + r^4)} \qquad l \geq 0 \tag{3.2.81}$$

Then from (3.2.80) we can write an expression for the energy in terms of the polar coordinates of the complex conjugate pole pair

$$r(0) = \frac{d_0^2(1+r^2)}{(1-r^2)(1-2r^2\cos 2\theta + r^4)} \tag{3.2.82}$$

The normalized autocorrelation is given by

$$\rho(l) = \frac{r^l\{\sin[(l+1)\theta] - r^2\sin[(l-1)\theta]\}}{(1+r^2)\sin\theta} \qquad l \geq 0 \tag{3.2.83}$$

which can be rewritten as

$$\rho(l) = \frac{1}{\cos\beta} r^l \cos(l\theta - \beta) \qquad l \geq 0 \tag{3.2.84}$$

where
$$\tan\beta = \frac{(1-r^2)\cos\theta}{(1+r^2)\sin\theta} \tag{3.2.85}$$

Therefore, $\rho(l)$ is a damped cosine wave with its maximum amplitude at the origin.

Spectrum. By setting the two poles equal to

$$p_1 = r_1 e^{j\theta_1} \qquad p_2 = r_2 e^{j\theta_2} \tag{3.2.86}$$

the spectrum of an AP(2) model can be written as

$$R(e^{j\omega}) = \frac{d_0^2}{[1 - 2r_1\cos(\omega - \theta_1) + r_1^2][1 - 2r_2\cos(\omega - \theta_2) + r_2^2]} \tag{3.2.87}$$

There are four cases of interest

Pole locations	Peak locations	**Type of** $R(e^{j\omega})$
$p_1 > 0,\ p_2 > 0$	$\omega = 0$	Low-pass
$p_1 < 0,\ p_2 < 0$	$\omega = \pi$	High-pass
$p_1 > 0,\ p_2 < 0$	$\omega = 0,\ \omega = \pi$	Stopband
$p_{1,2} = re^{\pm j\theta}$	$0 < \omega < \pi$	Bandpass

and they depend on the location of the poles on the complex plane.

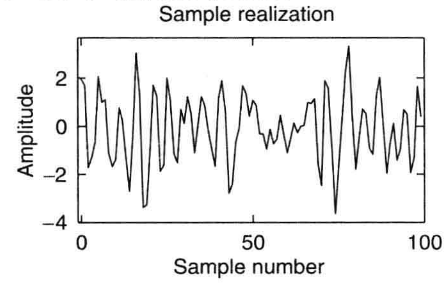
We concentrate on the fourth case of complex conjugate poles, which is of greatest interest. The other three cases are explored in Problem 3.15. The spectrum is given by

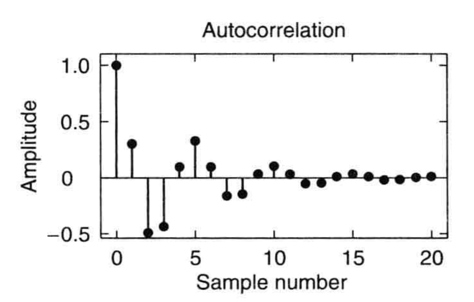
$$R(e^{j\omega}) = \frac{d_0^2}{[1 - 2r\cos(\omega - \theta) + r^2][1 - 2r\cos(\omega + \theta) + r^2]} \tag{3.2.88}$$

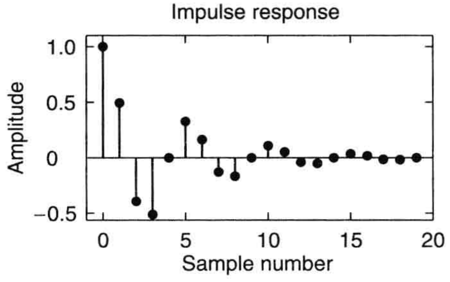
The peak of this spectrum can be shown to be located at a frequency ω_c, given by

$$\cos\omega_c = \frac{1+r^2}{2r}\cos\theta \tag{3.2.89}$$

Since $1 + r^2 > 2r$ for $r < 1$, and we have







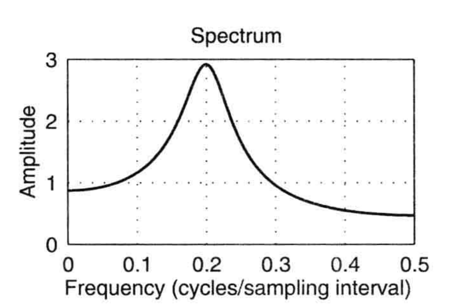


FIGURE 3.7
Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(2) model with complex conjugate poles.

$$\cos \omega_c > \cos \theta \tag{3.2.90}$$

the spectral peak is lower than the pole frequency for $0 < \theta < \pi/2$ and higher than the pole frequency for $\pi/2 < \theta < \pi$.

This behavior is illustrated in Figure 3.7 for an AP(2) model with $a_1 = -0.4944$, $a_2 = 0.64$, and $d_0 = 1$. The model has two complex conjugate poles with $r = 0.8$ and $\theta = \pm 2\pi/5$. The spectrum has a single peak and displays a passband type of behavior. The impulse response is a damped sine wave while the autocorrelation is a damped cosine. The typical realization of the output shows clearly a pseudoperiodic behavior that is explained by the shape of the autocorrelation and the spectrum of the model. We also notice that if the poles are complex conjugates, the autocorrelation has pseudoperiodic behavior.

Equivalent model descriptions. We now write explicit formulas for a_1 and a_2 in terms of the lattice parameters k_1 and k_2 and the autocorrelation coefficients. From the step-up and step-down recursions, we have

$$\begin{aligned} a_1 &= k_1(1+k_2) \\ a_2 &= k_2 \end{aligned} \tag{3.2.91}$$

and the inverse relations

$$\begin{aligned} k_1 &= \frac{a_1}{1+a_2} \\ k_2 &= a_2 \end{aligned} \tag{3.2.92}$$

From the Yule-Walker equations (3.2.18), we can write the two equations

$$\begin{aligned} a_1 r(0) + a_2 r(1) &= -r(1) \\ a_1 r(1) + a_2 r(0) &= -r(2) \end{aligned} \tag{3.2.93}$$

which can be solved for a_1 and a_2 in terms of $\rho(1)$ and $\rho(2)$

$$\begin{aligned} a_1 &= -\rho(1)\frac{1-\rho(2)}{1-\rho^2(1)} \\ a_2 &= \frac{\rho^2(1)-\rho(2)}{1-\rho^2(1)} \end{aligned} \tag{3.2.94}$$

or for $\rho(1)$ and $\rho(2)$ in terms of a_1 and a_2

$$\begin{aligned} \rho(1) &= -\frac{a_1}{1+a_2} \\ \rho(2) &= -a_1\rho(1) - a_2 = \frac{a_1^2}{1+a_2} - a_2 \end{aligned} \tag{3.2.95}$$

From the equations above, we can also write the relation and inverse relation between the coefficients k_1 and k_2 and the normalized autocorrelations $\rho(1)$ and $\rho(2)$ as

$$\begin{aligned} k_1 &= -\rho(1) \\ k_2 &= \frac{\rho^2(1)-\rho(2)}{1-\rho^2(1)} \end{aligned} \tag{3.2.96}$$

and

$$\begin{aligned} \rho(1) &= -k_1 \\ \rho(2) &= k_1(1+k_2) - k_2 \end{aligned} \tag{3.2.97}$$

The gain d_0 can also be written in terms of the other coefficients. From (3.2.20), we have

$$d_0^2 = r(0)[1 + a_1\rho(1) + a_2\rho(2)] \tag{3.2.98}$$

which can be shown to be equal to

$$d_0^2 = r(0)(1-k_1)(1-k_2) \tag{3.2.99}$$

Minimum-phase conditions. In (3.2.71), we have a set of conditions on a_1 and a_2 so that the AP(2) model is minimum-phase, and Figure 3.6 shows the corresponding *admissible* region for minimum-phase models. Similar relations and regions can be derived for the other types of parameters, as we will show below. In terms of k_1 and k_2, the AP(2) model is minimum-phase if

$$|k_1| < 1 \qquad |k_2| < 1 \tag{3.2.100}$$

This region is depicted in Figure 3.8(a). Shown also is the region that results in complex roots, which is specified by

$$0 < k_2 < 1 \tag{3.2.101}$$

$$k_1 < \frac{4k_2}{(1+k_2)^2} \tag{3.2.102}$$

Because of the correlation matching property of all-pole models, we can find a minimum-phase all-pole model for every positive definite sequence of autocorrelation values. Therefore, the admissible region of autocorrelation values coincides with the positive definite region. The positive definite condition is equivalent to having all the principal minors of the autocorrelation matrix in (3.2.30) be positive definite; that is, the corresponding determinants are positive. For $P = 2$, there are two conditions:

$$\det\begin{bmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{bmatrix} < 1 \qquad \det\begin{bmatrix} 1 & \rho(1) & \rho(2) \\ \rho(1) & 1 & \rho(1) \\ \rho(2) & \rho(1) & 1 \end{bmatrix} < 1 \tag{3.2.103}$$

These two conditions reduce to

$$|\rho(1)| < 1 \tag{3.2.104}$$

$$2\rho^2(1) - 1 < \rho(2) < 1 \tag{3.2.105}$$

which determine the admissible region shown in Figure 3.8(b). Conditions (3.2.105) can also be derived from (3.2.71) and (3.2.95). The first condition in (3.2.105) is equivalent to

$$\left|\frac{a_1}{1+a_2}\right| < 1 \tag{3.2.106}$$

which can be shown to be equivalent to the last two conditions in (3.2.71).

It is important to note that the region in Figure 3.8(b) is the admissible region for *any* positive definite autocorrelation, including the autocorrelation of mixed-phase signals. This is reasonable since the autocorrelation does not contain phase information and allows the signal to have minimum- and maximum-phase components. What we are claiming here, however, is that for every autocorrelation sequence in the positive definite region, we can find a minimum-phase all-pole model with the same autocorrelation values. Therefore, for this problem, the positive definite region is identical to the admissible minimum-phase region.

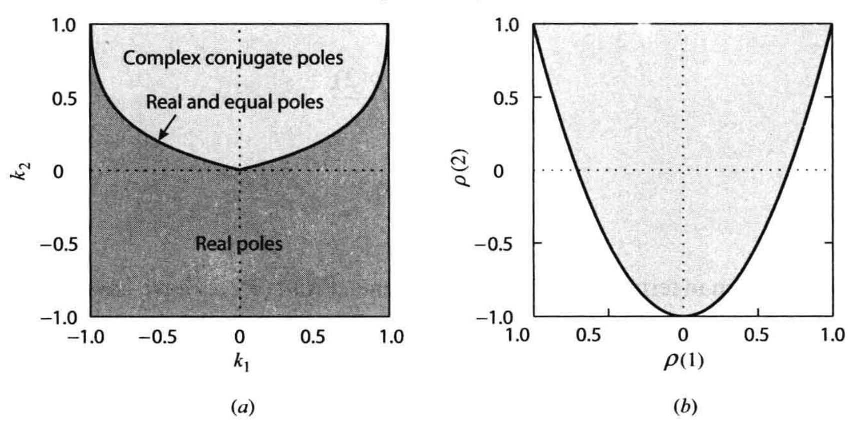


FIGURE 3.8
Minimum-phase and positive definiteness regions for the AP(2) model in the (a) (k_1, k_2) space and (b) ($\rho(1)$, $\rho(2)$) space.

3.3 All-Zero Models

In this section, we investigate the properties of the all-zero model. The output of the all-zero model is the weighted average of delayed versions of the input signal

$$x(n)=\sum_{k=0}^{Q} \mathrm{d}_k \omega(n-k) \tag{3.3.1}$$

where Q is the order of the model. The system function is

$$H(z)=D(z)=\sum_{k=0}^{Q} \mathrm{d}_k z^{-k} \tag{3.3.2}$$

The all-zero model can be implemented by using either a direct or a lattice structure. The conversion between the two sets of parameters can be done by using the step-up and step-down recursions described in Chapter 6 and setting $A(z)=D(z)$. Notice that the same set of parameters can be used to implement either an all-zero or an all-pole model by using a different structure.

3.3.1 Model Properties

We next provide a brief discussion of the properties of the all-zero model.

Impulse response. It can be easily seen that the AZ(Q) model is an FIR system with an impulse response

$$h(n)=\begin{cases} d_n & 0 \le n \le Q \\ 0 & \text{elsewhere} \end{cases} \tag{3.3.3}$$

Autocorrelation. The autocorrelation of the impulse response is given by

$$r_h(l)=\sum_{n=-\infty}^{\infty} h(n)h^*(n-l)=\begin{cases} \sum_{k=0}^{Q-l} \mathrm{d}_k \mathrm{d}_{k+l}^* & 0 \le l \le Q \\ 0 & l>Q \end{cases} \tag{3.3.4}$$

and

$$r_h^*(-l)=r_h(l) \qquad \text{all } l \tag{3.3.5}$$

We usually set $d_0=1$, which implies that

$$r_h(l)=d_l^*+d_1 d_{l+1}^*+\cdots+d_{Q-l}d_Q^* \qquad l=0,\ 1,\ \cdots,\ Q \tag{3.3.6}$$

hence, the normalized autocorrelation is

$$\rho_h(l)=\begin{cases} d_l^*+d_1 d_{l+1}^*+\cdots+d_{Q-l}d_Q^* / 1+|d_1|^2+\cdots+|d_Q|^2 & l=1,\ 2,\ \cdots,\ Q \\ 0 & l>Q \end{cases} \tag{3.3.7}$$

We see that the autocorrelation of an AZ(Q) model is zero for lags $|l|$ exceeding the order Q of the model. If $\rho_h(1)$, $\rho_h(2),\cdots,\rho_h(Q)$ are known, then the Q equations (3.3.7) can be solved for model parameters $d_1,\ d_2,\ \cdots,\ d_q$. However, unlike the Yule-Walker equations for the AP(P) model, which are linear, Equations (3.3.7) are nonlinear and their solution is quite complicated.

Spectrum. The spectrum of the AZ(Q) model is given by

$$R_h(\mathrm{e}^{\mathrm{j}\omega})=D(z)D(z^{-1})|_{z=\mathrm{e}^{\mathrm{j}\omega}}=|D(\mathrm{e}^{\mathrm{j}\omega})|^2=\sum_{l=-Q}^{Q} r_h(l)\mathrm{e}^{-\mathrm{j}\omega l} \tag{3.3.8}$$

which is basically a trigonometric polynomial.

Impulse train excitations. The response $\tilde{h}(n)$ of the AZ(Q) model to a periodic impulse train with period L is periodic with the same period, and its spectrum is a sampled version of (3.3.8) at multiples of $2\pi/L$. Therefore, to recover the autocorrelation $r_h(l)$ and the spectrum $R_h(\mathrm{e}^{\mathrm{j}\omega})$ from the autocorrelation or spectrum of $\tilde{h}(n)$, we should have $L \ge 2Q+1$ in order to avoid aliasing in the autocorrelation lag domain. Also, if $L>Q$, the impulse response $h(n)$, $0 \le n \le Q$, can be recovered from the response $\tilde{h}(n)$ (no time-domain aliasing) (see Problem 3.24).

Partial autocorrelation and lattice-ladder structures. The PACS of an AZ(Q) model is computed by fitting a series of AP(P) models for $P=1,\ 2,\ \cdots,$ to the autocorrelation sequence (3.3.7) of the AZ(Q) model. Since the AZ(Q) model is equivalent to an AP(∞) model, the PACS of an all-zero model has infinite extent and behaves as the autocorrelation sequence of an all-pole model. This is illustrated later for the low-order AZ(1) and AZ(2) models.

3.3.2 Moving-Average Models

A moving-average model is an AZ(Q) model with $d_0 = 1$ driven by white noise, that is,

$$x(n) = w(n) + \sum_{k=1}^{Q} d_k w(n-k) \tag{3.3.9}$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. The output $x(n)$ has zero mean and variance of

$$\sigma_x^2 = \sigma_w^2 \sum_{k=0}^{Q} |d_k|^2 \tag{3.3.10}$$

The autocorrelation and power spectrum are given by $r_x(l) = \sigma_w^2 r_h(l)$ and $R_x(e^{j\omega}) = \sigma_w^2 |D(e^{j\omega})|^2$, respectively. Clearly, observations that are more than Q samples apart are uncorrelated because the autocorrelation is zero after lag Q.

3.3.3 Lower-Order Models

To familiarize ourselves with all-zero models, we next investigate in detail the properties of the AZ(1) and AZ(2) models with real coefficients.

The first-order all-zero model: AZ(1). For generality, we consider an AZ(1) model whose system function is

$$H(z) = G(1 + d_1 z^{-1}) \tag{3.3.11}$$

The model is stable for any value of d_1 and minimum-phase for $-1 < d_1 < 1$. The autocorrelation is the inverse z-transform of

$$R_h(z) = H(z)H(z^{-1}) = G^2[d_1 z + (1 + d_1^2) + d_1 z^{-1}] \tag{3.3.12}$$

Hence, $r_h(0) = G^2(1 + d_1^2)$, $r_h(1) = r_h(-1) = G^2 d_1$, and $r_h(l) = 0$ elsewhere. Therefore, the normalized autocorrelation is

$$\rho_h(l) = \begin{cases} 1 & l = 0 \\ \dfrac{d_1}{1 + d_1^2} & l = \pm 1 \\ 0 & |l| \geq 2 \end{cases} \tag{3.3.13}$$

The condition $-1 < d_1 < 1$ implies that $|\rho_h(1)| \leq 1/2$ for a minimum-phase model. From $\rho_h(1) = d_1/(1 + d_1^2)$, we obtain the quadratic equation

$$\rho_h(1) d_1^2 - d_1 + \rho_h(1) = 0 \tag{3.3.14}$$

which has the following two roots:

$$d_1 = \frac{1 \pm \sqrt{1 - 4\rho_h^2(1)}}{2\rho_h(1)} \tag{3.3.15}$$

Since the product of the roots is 1, if d_1 is a root, then $1/d_1$ must also be a root. Hence, only one of these two roots can satisfy the minimum-phase condition $-1 < d_1 < 1$.

The spectrum is obtained by setting $z = e^{j\omega}$ in (3.3.12), or from (3.3.8)

$$R_h(e^{j\omega}) = G^2(1 + d_1^2 + 2d_1 \cos\omega) \tag{3.3.16}$$

The autocorrelation is positive definite if $R_h(e^{j\omega}) > 0$, which holds for all values of d_1. Note that if $d_1 > 0$, then $\rho_h(1) > 0$ and the spectrum has low-pass behavior (see Figure 3.9), whereas a high-pass spectrum is obtained when $d_1 < 0$ (see Figure 3.10).

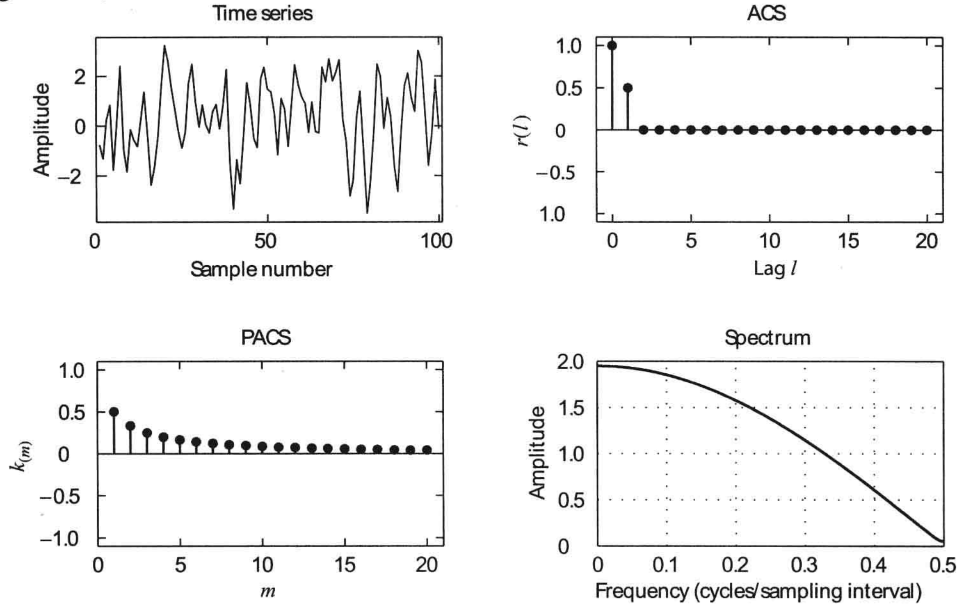


FIGURE 3.9

Sample realization of the output process, ACS, PACS, and spectrum of an AZ(1) model with $d_1 = 0.95$.

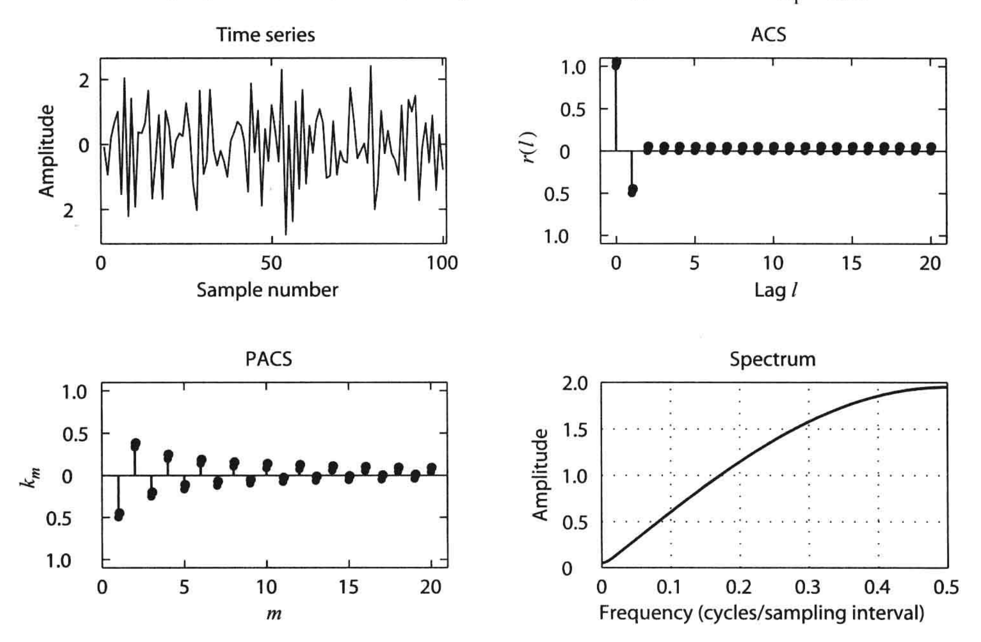


FIGURE 3.10

Sample realization of the output process, ACS, PACS, and spectrum of an AZ(1) model with $d_1 = -0.95$.

The first lattice parameter of the AZ(1) model is $k_1 = d_1$. The PACS can be obtained from the Yule-Walker equations by using the autocorrelation sequence (3.3.13). Indeed, after some algebra we obtain

$$k_m = \frac{(-d_1)^m (1-d_1^2)}{1-d_1^{2(m+1)}} \qquad m = 1,\ 2,\ \cdots,\ \infty \tag{3.3.17}$$

(see Problem 3.25). Notice the duality between the ACS and PACS of AP(1) and AZ(1) models.

Consider now the MA(1) real-valued process $x(n)$ generated by

$$x(n) = w(n) + bw(n-1)$$

where $\{w(n)\} \sim \mathrm{WN}(0, \sigma_w^2)$. Using $R_x(z) = \sigma_w^2 H(z) H^*(1/z^*)$, we obtain the PSD function

$$R_x(e^{j\omega}) = \sigma_\omega^2(1 + b^2 + 2b\cos\omega)$$

which has low-pass (high-pass) characteristics if $0 < b \le 1$ $(-1 \le b < 0)$. Since $\sigma_x^2 = r_x(0) = \sigma_\omega^2(1+b^2)$, we have (see Section 3.1.18)

$$\text{SFM}_x = \frac{\sigma_\omega^2}{\sigma_x^2} = \frac{1}{1+b^2} \tag{3.3.18}$$

which is maximum for $b = 0$ (white noise). The correlation matrix is banded Toeplitz (only a number of diagonals close to the main diagonal are nonzero)

$$\boldsymbol{R}_x = \sigma_\omega^2(1+b^2)\begin{bmatrix} 1 & b & 0 & \cdots & 0 \\ b & 1 & b & \cdots & 0 \\ 0 & b & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.3.19}$$

and its eigenvalues and eigenvectors are given by $\lambda_k = R_x(e^{j\omega_k})$, $q_n^{(k)} = \sin\omega_k n$, $\omega_k = \pi k/(M+1)$, where $k = 1, 2, \cdots, M$ (see Problem 3.30).

The second-order all-zero model: AZ(2). Now let us consider the second-order all-zero model. The system function of the AZ(2) model is

$$H(z) = G(1 + d_1 z^{-1} + d_2 z^{-2}) \tag{3.3.20}$$

The system is stable for all values of d_1 and d_2, and minimum-phase [see the discussion for the AP(2) model] if

$$\begin{aligned} -1 &< d_2 < 1 \\ d_2 - d_1 &> -1 \\ d_2 + d_1 &> -1 \end{aligned} \tag{3.3.21}$$

which is a triangular region identical to that shown in Figure 3.6. The normalized autocorrelation and the spectrum are

$$\rho_h(l) = \begin{cases} 1 & l = 0 \\ \dfrac{d_1(1+d_2)}{1+d_1^2+d_2^2} & l = \pm 1 \\ \dfrac{d_2}{1+d_1^2+d_2^2} & l = \pm 2 \\ 0 & |l| \ge 3 \end{cases} \tag{3.3.22}$$

and $R_h(e^{j\omega}) = G^2[(1+d_1^2+d_2^2) + 2d_1(1+d_2)\cos\omega + 2d_2\cos 2\omega]$ (3.3.23)

respectively.

FIGURE 3.11
Minimum-phase region in the autocorrelation domain for the AZ(2) model.

The minimum-phase region in the autocorrelation domain is shown in Figure 3.11 and is described by the equations

$$\begin{aligned} \rho(2) + \rho(1) &= -0.5 \\ \rho(2) - \rho(1) &= -0.5 \\ \rho^2(1) &= 4\rho(2)[1 - 2\rho(2)] \end{aligned} \tag{3.3.24}$$

derived in Problem 3.26. The formula for the PACS is quite involved. The important thing is the duality between the ACS and the PACS of AZ(2) and AP(2) models (see Problem 3.27).

3.4 Pole-Zero Models

We will focus on causal pole-zero models with a recursive input-output relationship given by

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + \sum_{k=0}^{Q} d_k \omega(n-k) \tag{3.4.1}$$

where we assume that $P > 0$ and $Q \geq 1$. The models can be implemented using either direct-form or lattice-ladder structures (Proakis and Manolakis 1996).

3.4.1 Model Properties

In this section, we present some of the basic properties of pole-zero models.

Impulse response. The impulse response of a causal pole-zero model can be written in recursive form from (3.4.1) as

$$h(n) = -\sum_{k=1}^{P} a_k h(n-k) + d_n \qquad n \geq 0 \tag{3.4.2}$$

where
$$d_n = 0 \qquad n > Q$$

and $h(n)$=0 for n<0. Clearly, this formula is useful if the model is stable. From (3.4.2), it is clear that

$$h(n) = -\sum_{k=1}^{P} a_k h(n-k) \qquad n \geq Q \tag{3.4.3}$$

so that the impulse response obeys the linear prediction equation for $n > Q$. Thus if we are given h(n), $0 \leq n \leq P+Q$, we can compute $\{a_k\}$ from (3.4.3) by using the P equations specified by $Q+1 \leq n \leq Q+P$. Then we can compute $\{d_k\}$ from {3.4.2}, using $0 \leq n \leq Q$. Therefore, the first $P+Q+1$ values of the impulse response completely specify the pole-zero model.

If the model is minimum-phase, the impulse response of the inverse model $h_I(n) = Z^{-1}\{A(z)/D(z)\}$, $d_0 = 1$ can be computed in a similar manner.

Autocorrelation. The complex spectrum of $H(z)$ is given by

$$R_h(z) = H(z)H^*\left(\frac{1}{z^*}\right) = \frac{D(z)D^*(1/z^*)}{A(z)A^*(1/z^*)} \triangleq \frac{R_d(z)}{R_a(z)} \tag{3.4.4}$$

where $R_d(z)$ and $R_a(z)$ are both finite two-sided polynomials. In a manner similar to the all-pole case, we can write a recursive relation between the autocorrelation, impulse response, and parameters of the model. Indeed, from (3.4.4) we obtain

$$A(z)R_h(z) = D(z)H^*\left(\frac{1}{z^*}\right) \tag{3.4.5}$$

Taking the inverse z-transform of (3.4.5) and noting that the inverse z-transform of $H(1/z^*)$ is $h^*(-n)$, we have

$$\sum_{k=0}^{P} a_k r_h(l-k) = \sum_{k=0}^{Q} d_k h^*(k-l) \qquad \text{for all } l \tag{3.4.6}$$

Since $h(n)$ is causal, we see that the right-hand side of (3.4.6) is zero for $l > Q$:

$$\sum_{k=0}^{P} a_k r_h(l-k) = 0 \qquad l > Q \tag{3.4.7}$$

Therefore, the autocorrelation of a pole-zero model obeys the linear prediction equation for $l > Q$.

Because the impulse response $h(n)$ is a function of a_k and d_k, the set of equations in (3.4.6) is nonlinear in terms of parameters a_k and d_k. However, (3.4.7) is linear in a_k; therefore, we can compute $\{a_k\}$ from (3.4.7), using the set of equations for $l = Q+1, \cdots, Q+P$, which can be written in matrix form as

$$\begin{bmatrix} r_h(Q) & r_h(Q-1) & \cdots & r_h(Q+P-1) \\ r_h(Q-1) & r_h(Q) & \cdots & r_h(Q+P-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_h(Q-P+1) & r_h(Q-P+2) & \cdots & r_h(Q) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = - \begin{bmatrix} r_h(Q-1) \\ r_h(Q-2) \\ \vdots \\ r_h(Q-P) \end{bmatrix} \tag{3.4.8}$$

or
$$\bar{\boldsymbol{R}}_h \boldsymbol{a} = -\bar{\boldsymbol{r}}_h \tag{3.4.9}$$

Here, $\bar{\boldsymbol{R}}_h$ is a non-Hermitian Toeplitz matrix, and the linear system (3.4.8) can be solved by using the algorithm of Trench (Trench 1964; Carayannis et al. 1981).

Even after we solve for $\boldsymbol{a}$, (3.4.6) continues to be nonlinear in d_k. To compute d_k, we use (3.4.4) to find $R_d(z)$

$$R_d(z) = R_a(z) R_h(z) \tag{3.4.10}$$

where the coefficients of $R_a(z)$ are given by

$$r_a(l) = \sum_{k=k_1}^{P-|l|} a_k a_{k+|l|}^* \qquad -P \le l \le P \tag{3.4.11}$$

From (3.4.10), $r_d(l)$ is the convolution of $r_a(l)$ with $r(l)$, given by

$$r_d(l) = \sum_{k=-P}^{P} r_a(k) r_h(l-k) \tag{3.4.12}$$

If $r(l)$ was originally the autocorrelation of a PZ(P,Q) model, then $r_d(l)$ in (3.4.12) will be zero for $|l| > Q$. Since $R_d(z)$ is specified, it can be factored into the product of two polynomials $D(z)$ and $D^*(1/z^*)$, where $D(z)$ is minimum-phase.

Therefore, we have seen that, given the values of the autocorrelation $r_h(l)$ of a PZ(P,Q) model in the range $0 \le l \le P+Q$, we can compute the values of the parameters $\{a_k\}$ and $\{d_k\}$ such that $H(z)$ is minimum-phase. Now, given the parameters of a pole-zero model, we can compute its autocorrelation as follows. Equation (3.4.4) can be written as

$$R_h(z) = R_a^{-1}(z) R_d(z) \tag{3.4.13}$$

where $R_a^{-1}(z)$ is the spectrum of the all-pole model $1/A(z)$, that is, $1/R_a(z)$. The coefficients of $R_a^{-1}(z)$ can be computed from $\{a_k\}$ by using (3.2.20) and (3.2.18). The coefficients of $R_d(z)$ are computed from (3.3.8). Then $R_h(z)$ is the convolution of the two autocorrelations thus computed, which is equivalent to multiplying the two polynomials in (3.4.13) and equating equal powers of z on both sides of the equation. Since $R_d(z)$ is finite, the summations used to obtain the coefficients of $R_h(z)$ are also finite.

EXAMPLE 3.4.1. Consider a signal that has autocorrelation values of $r_h(0) = 19$, $r_h(1) = 9$, $r_h(2) = -5$, and $r_h(3) = -7$. The parameters of the PZ(2, 1) model are found in the following manner. First form the equation from (3.4.8)

$$\begin{bmatrix} 9 & 19 \\ -5 & 9 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

which yields $a_1 = -1/2$, $a_2 = 1/2$. Then we compute the coefficients from (3.4.11), $r_a(0) = 3/2$, $r_a(\pm 1) = -3/4$, and $r_a(\pm 2) = 1/2$. Computing the convolution in (3.4.12) for $l \le Q = 1$, we obtain the following polynomial:

$$R_d(z) = 4z + 10 + 4z^{-1} = 4\left(1 + \frac{1}{2z^{-1}}\right)(z+2)$$

Therefore, $D(z)$ is obtained by taking the causal part, that is, $D(z) = 2[1 + 1/(2z^{-1})]$, and $d_1 = 1/2$.

Spectrum. The spectrum of $H(z)$ is given by

$$R_h(e^{j\omega}) = |H(e^{j\omega})|^2 = \frac{|D(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \tag{3.4.14}$$

Therefore, $R_h(e^{j\omega})$ can be obtained by dividing the spectrum of $D(z)$ by the spectrum of $A(z)$. Again, the FFT can be used to advantage in computing the numerator and denominator of (3.4.14). If the spectrum $R_h(e^{j\omega})$ of a PZ(P, Q) model is given, then the parameters of the (minimum-phase) model can be recovered by first computing the autocorrelation $r_h(l)$ as the inverse Fourier transform of $R_h(e^{j\omega})$ and then using the procedure outlined in the previous section to compute the sets of coefficients $\{a_k\}$ and $\{d_k\}$.

Partial autocorrelation and lattice-ladder structures. Since a PZ(P,Q) model is equivalent to an AP(∞) model, its PACS has infinite extent and behaves, after a certain lag, as the PACS of an all-zero model.

3.4.2 Autoregressive Moving-Average Models

The autoregressive moving-average model is a PZ(P,Q) model driven by white noise and is denoted by ARMA(P,Q). Again, we set $d_0 = 1$ and incorporate the gain into the variance (power) of the white noise excitation. Hence, a causal ARMA(P,Q) model is defined by

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + \omega(n) + \sum_{k=1}^{Q} d_k \omega(n-k) \tag{3.4.15}$$

where $\{\omega(n)\} \sim \text{WN}(0, \sigma_\omega^2)$. The ARMA(P,Q) model parameters are $\{\sigma_\omega^2,\ a_1,\ldots,a_P,\ d_1,\ldots,d_Q\}$. The output has zero mean and variance of

$$\sigma_x^2 = -\sum_{k=1}^{P} a_k r_x(k) + \sigma_\omega^2[1 + \sum_{k=1}^{Q} d_k h(k)] \tag{3.4.16}$$

where $h(n)$ is the impulse response of the model. The presence of $h(n)$ in (3.4.16) makes the dependence of σ_x^2 on the model parameters highly nonlinear. The autocorrelation of $x(n)$ is given by

$$\sum_{k=0}^{P} a_k r_x(l-k) = \sigma_\omega^2 \left[1 + \sum_{k=1}^{Q} d_k h(k-l) \right] \qquad \text{for all } l \tag{3.4.17}$$

and the power spectrum by

$$R_x(e^{j\omega}) = \sigma_\omega^2 \frac{|D(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \tag{3.4.18}$$

The significance of ARMA(P, Q) models is that they can provide more accurate representations than AR or MA models with the same number of parameters. *The ARMA model is able to combine the spectral peak matching of the AR model with the ability of the MA model to place nulls in the spectrum.*

3.4.3 The First-Order Pole-Zero Model: PZ(1, 1)

Consider the PZ(1, 1) model with the following system function

$$H(z) = G\frac{1 + d_1 z^{-1}}{1 + a_1 z^{-1}} \tag{3.4.19}$$

where d_1 and a_1 are real coefficients. The model is minimum-phase if

$$\begin{aligned} -1 < d_1 < 1 \\ -1 < a_1 < 1 \end{aligned} \tag{3.4.20}$$

which correspond to the rectangular region shown in Figure 3.12(a).

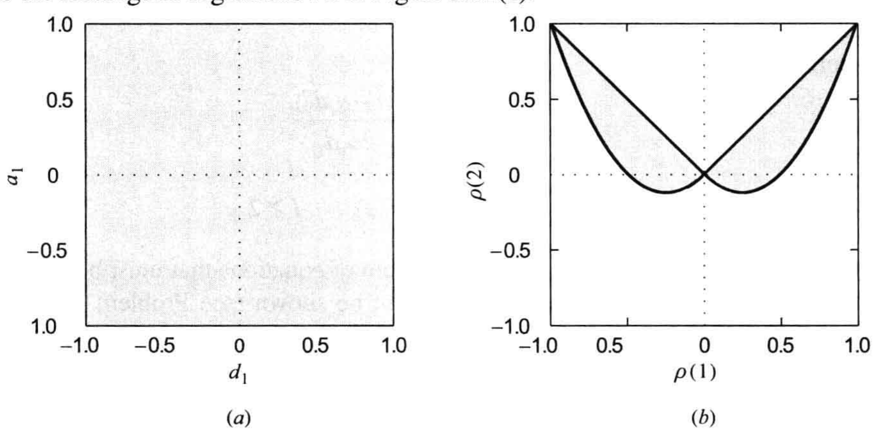


FIGURE 3.12
Minimum-phase and positive definiteness regions for the PZ(1,1) model in the (*a*) (d_1, a_1) space and (*b*) $(\rho(1), \rho(2))$ space.

For the minimum-phase case, the impulse responses of the direct and the inverse models are

$$h(n) = Z^{-1}\{H(z)\} = \begin{cases} 0 & n<0 \\ G & n=0 \\ G(-a_1)^{n-1}(d_1 - a_1) & n>0 \end{cases} \tag{3.4.21}$$

and

$$h_I(n) = Z^{-1}\left\{\frac{1}{H(z)}\right\} = \begin{cases} 0 & n<0 \\ G & n=0 \\ G(-d_1)^{n-1}(a_1 - d_1) & n>0 \end{cases} \tag{3.4.22}$$

respectively. We note that as the pole $p = -a_1$ gets closer to the unit circle, the impulse response decays more slowly and the model has "longer memory." The zero $z = -d_1$ controls the impulse response of the inverse model in a similar way. The PZ(1, 1) model is equivalent to the AZ(∞) model

$$x(n) = G\omega(n) + G\sum_{k=1}^{\infty} h(k)\omega(n-k) \tag{3.4.23}$$

or the AP(∞) model

$$x(n) = -\sum_{k=1}^{\infty} h_I(k)x(n-k) + G\omega(n) \tag{3.4.24}$$

If we wish to approximate the PZ(1, 1) model with a finite-order AZ(Q) model, the order Q required to achieve a certain accuracy increases as the pole moves closer to the unit circle. Likewise, in the case of an AP(P) approximation, better fits to the PZ(P, Q) model require an increased order P as the zero moves closer to the unit circle.

To determine the autocorrelation, we recall from (3.4.6) that for a causal model

$$r_h(l) = -a_1 r_h(l-1) + Gh(-l) + Gd_1 h(1-l) \qquad \text{all } l \tag{3.4.25}$$

or

$$\begin{aligned} r_h(0) &= -a_1 r_h(1) + G + Gd_1(d_1 - a_1) \\ r_h(1) &= -a_1 r_h(0) + Gd_1 \\ r_h(l) &= -a_1 r_h(l-1) \qquad l \ge 2 \end{aligned} \tag{3.4.26}$$

Solving the first two equations for $r_h(0)$ and $r_h(1)$, we obtain

$$r_h(0) = G\frac{1 + d_1^2 - 2a_1 d_1}{1 - a_1^2} \tag{3.4.27}$$

and

$$r_h(1) = G\frac{(d_1 - a_1)(1 - a_1 d_1)}{1 - a_1^2} \tag{3.4.28}$$

The normalized autocorrelation is given by

$$\rho_h(1) = \frac{(d_1 - a_1)(1 - a_1 d_1)}{1 + d_1^2 - 2a_1 d_1} \tag{3.4.29}$$

and

$$\rho_h(l) = (-a_1)^{l-1}\rho_h(l-1) \qquad l \ge 2 \tag{3.4.30}$$

Note that given $\rho_h(1)$ and $\rho_h(2)$, we have a nonlinear system of equations that must be solved to obtain a_1 and d_1. By using Equations (3.4.20), (3.4.29), and (3.4.30), it can be shown (see Problem 3.28) that the PZ(1, 1) is minimum-phase if the ACS satisfies the conditions

$$\begin{aligned} |\rho(2)| &< |\rho(1)| \\ \rho(2) &> \rho(1)[2\rho(1) + 1] \qquad \rho(1) < 0 \\ \rho(2) &> \rho(1)[2\rho(1) - 1] \qquad \rho(1) > 0 \end{aligned} \tag{3.4.31}$$

which correspond to the admissible region shown in Figure 3.12(b).

3.4.4 Summary and Dualities

Table 3.1 summarizes the key properties of all-zero, all-pole, and pole-zero models. These properties help to identify models for empirical discrete-time signals. Furthermore, the table shows the duality between AZ and AP models.

More specifically, we see that
1. An invertible AZ(Q) model is equivalent to an AP(∞) model. Thus, it has a finite-extent autocorrelation and an infinite-extent partial autocorrelation.
2. A stable AP(P) model is equivalent to an AZ(∞) model. Thus, it has an infinite-extent autocorrelation and a finite-extent partial autocorrelation.
3. The autocorrelation of an AZ(Q) model behaves as the partial autocorrelation of an AP(P) model, and vice versa.
4. The spectra of an AP(P) model and an AZ(Q) model are related through an inverse relationship.

Table 3.1.

Summary of all-pole, all-zero, and pole-zero model properties

Model	AP(*P*)	AZ(*Q*)	PZ(*P*,*Q*)
Input-output description	$x(n)+\sum_{k=1}^{P} a_k x(n-k)=\omega(n)$	$x(n)=d_0\omega(n)+\sum_{k=1}^{Q} d_k\omega(n-k)$	$x(n)+\sum_{k=1}^{P} a_k x(n-k)$ $=d_0\omega(n)+\sum_{k=1}^{Q} d_k\omega(n-k)$
System function	$H(z)=1/A(z)=d_0/1+\sum_{k=1}^{P} a_k z^{-k}$	$H(z)=D(z)=d_0+\sum_{k=1}^{Q} d_k z^{-k}$	$H(z)=D(z)/A(z)$
Recursive representation	Finite summation	Infinite summation	Infinite summation
Nonrecursive representation	Infinite summation	Finite summation	Infinite summation
Stablity conditions	Poles inside unit circle	Always	Poles inside unit circle
Invertiblity conditions	Always	Zeros inside unit circle	Zeros inside unit circle
Autocorrelation sequence	Infinite duration (damped exponentials and/or sine waves)	Finite duration	Infinite duration (damped exponentials and/or sine waves after $Q-P$ lags) Tails off
Partial autocorrelation	Tails off Finite duration	Cuts off Infinite duration (damped exponentials and/or sine waves)	Infinite duration (dominated by damped exponentials and/or sine waves after $Q-P$ lags)
Spectrum	Cuts off Good peak matching	Tails off Good "notch" matching	Tails off Good peak and valley matching

These dualities and properties have been shown and illustrated for low-order models in the previous sections.

3.5 Summary

In this chapter we introduced the class of pole-zero signal models and discussed their properties. Each model consists of two components: an excitation source and a system. In our treatment, we emphasized that the properties of a signal model are shaped by the properties of both components; and we tried, whenever possible, to attribute each property to its originator. Thus, for uncorrelated random inputs, which by definition are the excitations for ARMA models, the second-order moments of the signal model and its minimum-phase characteristics are completely determined by the system.

We provided a detailed description of the autocorrelation, power spectrum density of all AZ, AP, and PZ models for the general case and for first- and second-order models. An understanding of these properties is very important for model selection in practical applications.

Problems

3.1 Show that a second-order pole p_i contributes the term $np_i^n u(n)$ and a third-order pole the terms $np_i^n u(n)+n^2 p_i^n u(n)$ to the

impulse response of a causal PZ model. The general case is discussed in Oppenheim et al. (1997).

3.2 Consider a zero-mean random sequence $x(n)$ with PSD

$$R_x(e^{j\omega}) = \frac{5+3\cos\omega}{17+8\cos\omega}$$

(*a*) Determine the innovations representation of the process $x(n)$.

(*b*) Find the autocorrelation sequence $r_x(l)$.

3.3 We want to generate samples of a Gaussian process with autocorrelation $r_x(l) = (\frac{1}{2})^{|l|} + (-\frac{1}{2})^{|l|}$ for all l.

(*a*) Find the difference equation that generates the process $x(n)$ when excited by $\omega(n) \sim \mathrm{WGN}(0,1)$.

(*b*) Generate $N = 1000$ samples of the process and estimate the pdf, using the histogram and the normalized autocorrelation $\rho_x(l)$ using $\hat{\rho}_x(l)$ [see Section (1.2.1)].

(*c*) Check the validity of the model by plotting on the same graph (i) the true and estimated pdf of $y(n)$ and (ii) the true and estimated autocorrelation.

3.4 Compute and compare the autocorrelations of the following processes:

(*a*) $x_1(n) = \omega(n) + 0.3\omega(n-1) - 0.4\omega(n-1)$ and

(*b*) $x_1(n) = \omega(n) - 1.2\omega(n-1) - 1.6\omega(n-1)$ where $\omega(n) \sim \mathrm{WGN}(0,1)$.

Explain your findings.

3.5 Compute and plot the impulse response and the magnitude response of the systems $H(z)$ and $H_N(z)$ in Example 3.2.1 for $a = 0.7, 0.95$ and $N = 8,\ 16,\ 64$. Investigate how well the all-zero systems approximate the single-pole system.

3.6 Prove Equation (3.2.25) by writing explicitly Equation (3.2.23) and rearranging terms. Then show that the coefficient matrix $\mathbf{A}$ can be written as the sum of a triangular Toeplitz matrix and a triangular Hankel matrix (recall that a matrix $\mathbf{H}$ is Hankel if the matrix $\mathbf{JHJ}^H$ is Toeplitz).

3.7 Use the Yule-Walker equations to determine the autocorrelation and partial autocorrelation coefficients of the following AR models, assuming that $\omega(n) \sim \mathrm{WN}(0,1)$.

(*a*) $x(n) = 0.5x(n-1) + \omega(n)$.

(*b*) $x(n) = 1.5x(n-1) - 0.6x(n-2) + \omega(n)$.

What is the variance σ_x^2 of the resulting process?

3.8 Given the AR process $x(n) = x(n-1) - 0.5x(n-2) + \omega(n)$, complete the following tasks.

(*a*) Determine $\rho_x(1)$.

(*b*) Using $\rho_x(0)$ and $\rho_x(1)$, compute $\{\rho_x(l)\}_2^{15}$ by the corresponding difference equation.

(*c*) Plot $\rho_x(l)$ and use the resulting graph to estimate its period.

(*d*) Compare the period obtained in part (*c*) with the value obtained using the PSD of the model. (*Hint:* Use the frequency of the PSD peak.)

3.9 Given the parameters $d_0, a_1, a_2,$ and a_3 of an AP(3) model, compute its ACS analytically and verify your results, using the values in Example 3.2.3 (*Hint:* Use Cramer's rule.)

3.10 Consider the following AP(3) model: $x(n) = 0.98x(n-3) + \omega(n)$, where $\omega(n) \sim \mathrm{WGN}(0,1)$.

(*a*) Plot the PSD of $x(n)$ and check if the obtained process is going to exhibit a pseudoperiodic behavior.

(*b*) Generate and plot 100 samples of the process. Does the graph support the conclusion of part (*a*)? If yes, what is the period?

(*c*) Compute and plot the PSD of the process $y(n) = \frac{1}{3}[x(n-1) + x(n) + x(n+1)]$.

(*d*) Repeat part (*b*) and explain the difference between the behavior of processes $x(n)$ and $y(n)$.

3.11 Consider the following AR(2) models: (i) $x(n) = 0.6x(n-1) + 0.3x(n-2) + \omega(n)$ and (ii) $x(n) = 0.8x(n-1) - 0.5x(n-2) + \omega(n)$, where $\omega(n) \sim \mathrm{WGN}(0,1)$.

(*a*) Find the general expression for the normalized autocorrelation sequence $\rho(l)$, and determine σ_x^2.

(*b*) Plot $\{\rho(l)\}_0^{15}$ and check if the models exhibit pseudoperiodic behavior.

(*c*) Justify your answer in part (*b*) by plotting the PSD of the two models.

3.12 (*a*) Derive the formulas that express the PACS of an AP(3) model in terms of its ACS, using the Yule-Walker equations and Cramer's rule.

(*b*) Use the obtained formulas to compute the PACS of the AP(3) model in Example 3.2.3.

(*c*) Check the results in part (*b*) by recomputing the PACS, using the algorithm of Levinson-Durbin.

3.13 Show that the spectrum of any PZ model with real coefficients has zero slope at $\omega = 0$ and $\omega = \pi$.

3.14 Derive Equations (3.2.71) describing the minimum-phase region of the AP(2) model, starting from the conditions

(*a*) $|p_1| < 1$, $|p_2| < 1$ and

(*b*) $|k_1| < 1$, $|k_2| < 1$.

3.15 (*a*) Show that the spectrum of an AP(2) model with real poles can be obtained by the cascade connection of two AP(1) models with

real coefficients.

(*b*) Compute and plot the impulse response, ACS, PACS, and spectrum of the AP models with $p_1 = 0.6$, $p_2 = -0.9$, and $p_1 = p_2 = 0.9$.

3.16 Prove Equation (3.2.89) and demonstrate its validity by plotting the spectrum (3.2.88) for various values of r and θ.

3.17 Prove that if the AP(P) model $A(z)$ is minimum-phase, then

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\log\frac{1}{|A(\mathrm{e}^{\mathrm{j}\omega})|^2}\mathrm{d}\omega = 0$$

3.18 (*a*) Prove Equations (3.2.101) and (3.2.102) and recreate the plot in Figure 3.8(*a*).

(*b*) Determine and plot the regions corresponding to complex and real poles in the autocorrelation domain by recreating Figure 3.8(*b*).

3.19 Consider an AR(2) process $x(n)$ with $d_0 = 1$, $a_1 = -1.6454$ $a_2 = 0.9025$, and $\omega(n) \sim \mathrm{WGN}(0,1)$.

(*a*) Generate 100 samples of the process and use them to estimate the ACS $\hat{\rho}_x(l)$, using Equation (1.2.1).

(*b*) Plot and compare the estimated and theoretical ACS values for $0 \le l \le 10$.

(*c*) Use the estimated values of $\hat{\rho}_x(l)$ and the Yule-Walker equations to estimate the parameters of the model. Compare the estimated with the true values, and comment on the accuracy of the approach.

(*d*) Use the estimated parameters to compute the PSD of the process. Plot and compare the estimated and true PSDs of the process.

(*e*) Compute and compare the estimated with the true PACS.

3.20 Find a minimum-phase model with autocorrelation $\rho(0) = 1$, $\rho(\pm 1) = 0.25$, and $\rho(l) = 0$ for $|l| \ge 2$.

3.21 Consider the MA(2) model $x(n) = \omega(n) - 0.1\omega(n-1) + 0.2\omega(n-2)$.

(*a*) Is the process $x(n)$ stationary? Why?

(*b*) Is the model minimum-phase? Why?

(*c*) Determine the autocorrelation and partial autocorrelation of the process.

3.22 Consider the following ARMA models: (i) $x(n) = 0.6x(n-1) + \omega(n) - 0.9\omega(n-1)$ and (ii) $x(n) = 1.4x(n-1) - 0.6x(n-2) + \omega(n) - 0.8\omega(n-1)$.

(*a*) Find a general expression for the autocorrelation $\rho(l)$.

(*b*) Compute the partial autocorrelation k_m for $m = 1,\ 2,\ 3$.

(*c*) Generate 100 samples from each process, and use them to estimate $\{\hat{\rho}(l)\}_0^{20}$ using Equation (1.2.1).

(*d*) Use $\hat{\rho}(l)$ to estimate $\{\hat{k}_m\}_1^{20}$.

(*e*) Plot and compare the estimates with the theoretically obtained values.

3.23 Determine the coefficients of a PZ(2,1) model with autocorrelation values $r_h(0) = 19$, $r_h(1) = 9$, $r_h(2) = -5$, and $r_h(3) = -7$.

3.24 (*a*) Show that the impulse response of an AZ(Q) model can be recovered from its response $h(n)$ to a periodic train with period L if $L > Q$.

(*b*) Show that the ACS of an AZ(Q) model can be recovered from the ACS or spectrum of $h(n)$ if $L \ge 2Q+1$.

3.25 Prove Equation (3.3.17) and illustrate its validity by computing the PACS of the model $H(z) = 1 - 0.8z^{-1}$.

3.26 Prove Equations (3.3.24) that describe the minimum-phase region of the AZ(2) model.

3.27 Consider an AZ(2) model with $d_0 = 2$ and zeros $z_{1,2} = 0.95\mathrm{e}^{\pm \mathrm{j}\pi/3}$.

(*a*) Compute and plot $N = 100$ output samples by exciting the model with the process $\omega(n) \sim \mathrm{WGN}(0,1)$.

(*b*) Compute and plot the ACS, PACS, and spectrum of the model.

(*c*) Repeat parts (*a*) and (*b*) by assuming that we have an AP(2) model with poles at $p_{1,2} = 0.95\mathrm{e}^{\pm \mathrm{j}\pi/3}$.

(*d*) Investigate the duality between the ACS and PACS of the two models.

3.28 Prove Equations (3.4.31) and use them to reproduce the plot shown in Figure 3.12(*b*). Indicate which equation corresponds to each curve.

3.29 Determine the spectral flatness measure of the following processes:

(*a*) $x(n) = a_1x(n-1) + a_2x(n-2) + \omega(n)$ and

(*b*) $x(n) = \omega(n) + b_1\omega(n-1) + b_2\omega(n-2)$, where $\omega(n)$ is a white noise sequence.

3.30 Consider a zero-mean wide-sense stationary (WSS) process $x(n)$ with PSD $R_x(\mathrm{e}^{\mathrm{j}\omega})$ and an $M \times M$ correlation matrix with eigenvalues $\{\lambda_k\}_1^M$. Szegö's theorem (Grenander and Szegö 1958) states that if $g(\cdot)$ is a continuous function, then

$$\lim_{M\to\infty}\frac{g(\lambda_1)+g(\lambda_2)+\cdots+g(\lambda_M)}{M} = \frac{1}{2\pi}\int_{-\pi}^{\pi} g[R_x(\mathrm{e}^{\mathrm{j}\omega})]\mathrm{d}\omega$$

Using this theorem, show that

$$\lim_{M\to\infty}(\det \boldsymbol{R}_x)^{1/M} = \exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln[R_x(\mathrm{e}^{\mathrm{j}\omega})]\mathrm{d}\omega\right\}$$

3.31 Consider two linear random processes with system functions

(i) $H(z)=1-0.81z^{-1}-0.4z^{-2}/(1-z^{-1})^2$ and (ii) $H(z)=1-0.5z^{-1}/1-z^{-1}$

(*a*) Find a difference equation that leads to a numerically stable simulation of each process.

(*b*) Generate and plot 100 samples from each process, and look for indications of nonstationarity in the obtained records.

(*c*) Compute and plot the second difference of (i) and the first difference of (ii). Comment about the stationarity of the obtained records.

3.32 Generate and plot 100 samples for each of the linear processes with system functions

(*a*) $H(z)=\dfrac{1}{(1-z^{-1})(1-0.9z^{-1})}$

(*b*) $H(z)=\dfrac{1-0.5z^{-1}}{(1-z^{-1})(1-0.9z^{-1})}$

and then estimate and examine the values of the ACS $\{\hat{\rho}(l)\}_0^{20}$ and the PACS $\{\hat{k}_m\}_1^{20}$.

3.33 Consider the process $y(n)=d_0+d_1n+d_2n^2+x(n)$, where $x(n)$ is a stationary process with known autocorrelation $r_x(l)$.

(*a*) Show that the process $y^{(2)}(n)$ obtained by passing $y(n)$ through the filter $H(z)=(1-z^{-1})^2$ is stationary.

(*b*) Express the autocorrelation $r_y^{(2)}(l)$ of $y^{(2)}(n)$ in terms of $r_x(l)$. *Note:* This process is used in practice to remove quadratic trends from data before further analysis.

CHAPTER 4

Nonparametric Power Spectrum Estimation

The essence of frequency analysis is the representation of a signal as a superposition of sinusoidal components. In theory, the exact form of this decomposition (spectrum) depends on the assumed signal model. In Chapters 2 we discussed the mathematical tools required to define and compute the spectrum of signals described by stochastic models. In practical applications, where only a finite segment of a signal is available, we cannot obtain a complete description of the adopted signal model. Therefore, we can only compute an approximation (estimate) of the spectrum of the adopted signal model ("true" or theoretical spectrum). The quality of the estimated spectrum depends on

- How well the assumed signal model represents the data.
- What values we assign to the unavailable signal samples.
- Which spectrum estimation method we use.

Clearly, meaningful application of spectrum estimation in practical problems requires sufficient a priori information, understanding of the signal generation process, knowledge of theoretical concepts, and experience.

In this chapter we discuss the most widely used correlation and spectrum estimation methods, as well as their properties, implementation, and application to practical problems. We discuss only *nonparametric* techniques that do *not* assume a particular functional form, but allow the form of the estimator to be determined *entirely* by the data. These methods are based on the discrete Fourier transform of either the signal segment or its autocorrelation sequence. In contrast, parametric methods assume that the available signal segment has been generated by a specific parametric model (e.g., a pole-zero or harmonic model). Since the choice of an inappropriate signal model will lead to erroneous results, the successful application of parametric techniques, without sufficient a priori information, is very difficult in practice. These methods are discussed in Chapter 8.

We begin this chapter with an introductory discussion on the purpose of, and the DSP approach to, spectrum estimation. We explore various errors involved in the estimation of finite-length data records (i.e., based on partial information). Section 4.3 is the main section of this chapter in which we discuss various nonparametric approaches to the power spectrum estimation of stationary random signals. The computation of auto and cross-spectra using Thomson's multiple windows (or multitapers) is discussed in Section 4.5. Finally, in Section 4.6 we summarize important topics and concepts from this chapter. A classification of the various spectral estimation methods that are discussed in this book is provided in Figure 4.1.

4.1 Spectral Analysis of Deterministic Signals

If we adopt a deterministic signal model, the mathematical tools for spectral analysis are the Fourier series and the Fourier transforms. It should be stressed at this point that applying any of these tools requires that the signal values in the entire time interval from $-\infty$ to $+\infty$ be available. If it is known a priori that a signal is periodic, then only one period is needed. The rationale for defining and studying various spectra for deterministic signals is threefold. First, we note that every realization (or sample function) of a stochastic process is a deterministic function. Thus we can use the Fourier series and transforms to compute a spectrum for stationary processes. Second, deterministic functions and sequences are used in many aspects of the study of stationary processes, for example, the autocorrelation sequence, which is a deterministic sequence. Third, the various spectra that can be defined for deterministic signals can be used to summarize important features of stationary processes.

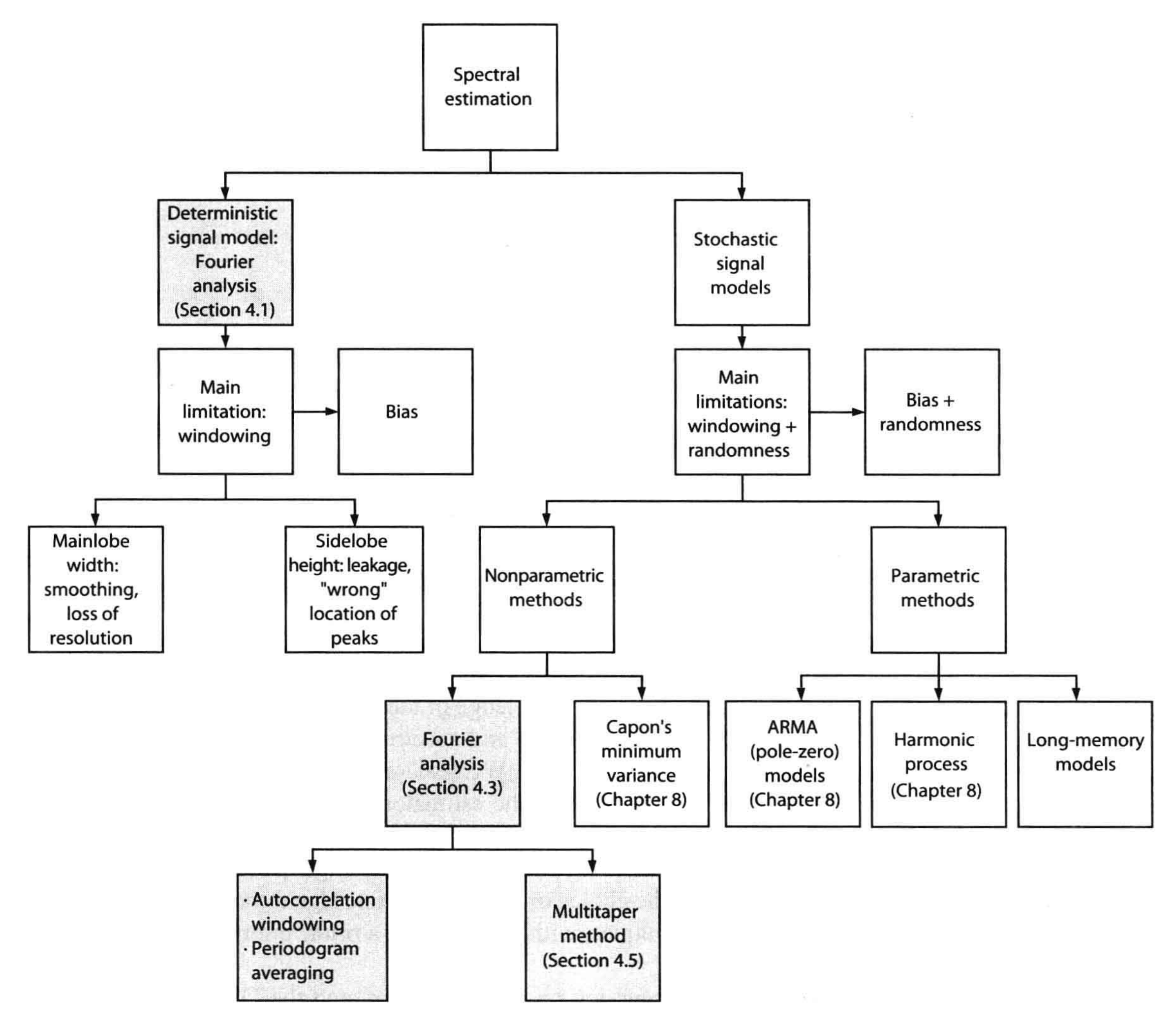


FIGURE 4.1
Classification of various spectrum estimation methods.

Most practical applications of spectrum estimation involve continuous-time signals. For example, in speech analysis we use spectrum estimation to determine the pitch of the glottal excitation and the formants of the vocal tract (Rabiner and Schafer 1978). In electroencephalography, we use spectrum estimation to study sleep disorders and the effect of medication on the functioning of the brain (Duffy, Iyer, and Surwillo 1989). Another application is in Doppler radar, where the frequency shift between the transmitted and the received waveform is used to determine the radial velocity of the target (Levanon 1988).

The numerical computation of the spectrum of a continuous-time signal involves three steps:

1. Sampling the continuous-time signal to obtain a sequence of samples.
2. Collecting a finite number of contiguous samples (data segment or block) to use for the computation of the spectrum. This operation, which usually includes weighting of the signal samples, is known as *windowing*, or *tapering*.
3. Computing the values of the spectrum at the desired set of frequencies. This step is usually implemented using some efficient implementation of the DFT.

The above processing steps, which are necessary for DFT-based spectrum estimation, are shown in Figure 4.2. The continuous-time signal is first processed through a low-pass (antialiasing) filter and then sampled to obtain a discrete-time signal. Data samples of frame length N with *frame overlap* N_0 are selected and then conditioned using a window. Finally, a suitable-length DFT of the windowed data is taken as an estimate of its spectrum, which is then analyzed. In this section, we discuss in detail the effects of each of these operations on the accuracy of the computed spectrum. The understanding of the implications of these effects is very important in all practical applications of

spectrum estimation.

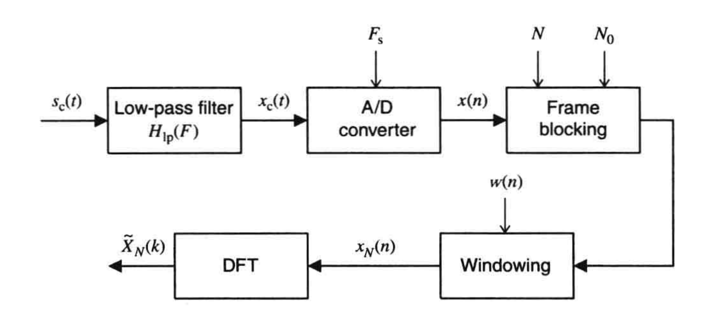


FIGURE 4.2
DFT-based Fourier analysis system for continuous-time signals.

4.1.1 *Effect of Signal Sampling*

The continuous-time signal $x_c(t)$, whose spectrum we seek to estimate, is first passed through a low-pass filter, also known as an *antialiasing* filter $H_{lp}(F)$, in order to minimize the aliasing error after sampling. The antialiased signal $x_c(t)$ is then sampled through an analog-to-digital converter[1] (ADC) to produce the discrete-time sequence $x(n)$, that is,

$$x(n) = x_c(t)\,|_{t=n/F_s} \tag{4.1.1}$$

From the sampling theorem, we have

$$X(e^{j2\pi F/F_s}) = F_s \sum_{l=-\infty}^{\infty} X_c(F - lF_s) \tag{4.1.2}$$

where $X_c(F) = H_{lp}(F)X_c(F)$. We note that the spectrum of the discrete-time signal $x(n)$ is a periodic replication of $X_c(F)$. Overlapping of the replicas $X_c(F - lF_s)$ results in aliasing. Since any practical antialiasing filter does not have infinite attenuation in the stopband, some nonzero overlap of frequencies higher than $F_s/2$ should be expected within the band of frequencies of interest in $x(n)$. These aliased frequencies give rise to the aliasing error, which, in any practical signal, is unavoidable. It can be made negligible by a properly designed antialiasing filter $H_{lp}(F)$.

4.1.2 *Windowing, Periodic Extension, and Extrapolation*

In practice, we compute the spectrum of a signal by using a finite-duration segment. The reason is threefold:
1. The spectral composition of the signal changes with time. or
2. We have only a finite set of data at our disposal. or
3. We wish to keep the computational complexity to an acceptable level.

Therefore, it is necessary to partition $x(n)$ into blocks (or *frames*) of data prior to processing. This operation is called *frame blocking*, and it is characterized by two parameters: the *length* of frame N and the *overlap* between frames N_0 (see Figure 4.2). Therefore, the central problem in practical frequency analysis can be stated as follows:

Determine the spectrum of a signal $x(n), -\infty < n < \infty$, from its values in a finite interval $0 \le n \le N-1$, that is, from a finite-duration segment.

Since $x(n)$ is unknown for $n < 0$ and $n \ge N$, we cannot say, without having sufficient a priori information, whether the signal is periodic or aperiodic. If we can reasonably assume that the signal is periodic with fundamental period N, we can easily determine its spectrum by computing its Fourier series, using the DFT.

However, in most practical applications, we cannot make this assumption because the available block of data could be either part of the period of a periodic signal or a segment from an aperiodic signal. In such cases, the spectrum of the signal *cannot* be determined without *assigning* values to the signal samples outside the available interval. There are three ways to deal with this issue:

1. *Periodic extension.* We assume that $x(n)$ is periodic with period N, that is, $x(n) = x(n+N)$ for all n, and we compute its Fourier series, using the DFT.

[1]We will ignore the quantization of discrete-time signals.

2. *Windowing.* We assume that the signal is zero outside the interval of observation, that is, $x(n) = 0$ for $n < 0$ and $n \geq N$. This is equivalent to multiplying the signal with the rectangular window

$$w_R(n) \triangleq \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{elsewhere} \end{cases} \tag{4.1.3}$$

The resulting sequence is aperiodic, and its spectrum is obtained by the discrete-time Fourier transform (DTFT).

3. *Extrapolation.* We use a priori information about the signal to extrapolate (i.e., determine its values for $n < 0$ and $n \geq N$) outside the available interval and then determine its spectrum by using the DTFT.

Periodic extension and windowing can be considered the simplest forms of extrapolation. It should be obvious that a successful extrapolation results in better spectrum estimates than periodic extension or windowing. Periodic extension is a straightforward application of the DFT, whereas extrapolation requires some form of a sophisticated signal model. As we shall see, most of the signal modeling techniques discussed in this book result in some kind of extrapolation. We first discuss, in the next section, the effect of spectrum sampling as imposed by the application of DFT (and its side effect—the periodic extension) before we provide a detailed analysis of the effect of windowing.

4.1.3 Effect of Spectrum Sampling

In many real-time spectrum analyzers, as illustrated in Figure 4.2, the spectrum is computed (after signal conditioning) by using the DFT. The computation samples the continuous spectrum at equispaced frequencies. Theoretically, if the number of DFT samples is greater than or equal to the frame length N, then the exact continuous spectrum (based on the given frame) can be obtained by using the frequency-domain reconstruction (Oppenheim and Schafer 1989; Proakis and Manolakis 1996). This reconstruction, which requires a periodic sinc function [defined in (4.1.9)], is not a practical function to implement, especially in real-time applications. Hence a simple linear interpolation is used for plotting or display purposes. This linear interpolation can lead to misleading results even though the computed DFT sample values are correct. It is possible that there may not be a DFT sample precisely at a frequency where a peak of the DTFT is located. In other words, the DFT spectrum misses this peak, and the resulting linearly interpolated spectrum provides the wrong location and height of the DTFT spectrum peak. This error can be made smaller by sampling the DTFT spectrum at a finer grid, that is, by increasing the size of the DFT. The denser spectrum sampling is implemented by an operation called *zero padding* and is discussed later in this section.

Another effect of the application of DFT for spectrum calculations is the periodic extension of the sequence in the time domain. It follows that the N-point DFT

$$\tilde{X}(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn} \tag{4.1.4}$$

is periodic with period N. This should be expected given the relationship of the DFT to the Fourier transform or the Fourier series of discrete-time signals, which are periodic in ω with period 2π. A careful look at the inverse DFT

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{j(2\pi/N)kn} \tag{4.1.5}$$

reveals that $x(n)$ is also periodic with period N. This is a somewhat surprising result since no assumption about the signal $x(n)$ outside the interval $0 \leq n \leq N-1$ has been made. However, this periodicity in the time domain can be easily justified by recalling that sampling in the time domain results in a periodicity in the frequency domain, and vice versa.

To understand these effects of spectrum sampling, consider the following example in which a continuous-time sinusoidal signal is sampled and then is truncated by a rectangular window before its DFT is performed.

EXAMPLE 4.1.1. A continuous-time signal $x_c(t) = 2\cos 2\pi t$ is sampled with a sampling frequency of $F_s = 1/T = 10$ samples per second, to obtain the sequence $x(n)$. It is windowed by an N-point rectangular window $w_R(n)$ to obtain the sequence $x_N(n)$. Determine and plot $|\tilde{X}_N(k)|$, the magnitude of the DFT of $x_N(n)$, for (a) $N = 10$ and (b) $N = 15$. Comment on the shapes of these plots.

Solution. The discrete-time signal $x(n)$ is a sampled version of $x_c(t)$ and is given by

$$x(n) = x_c(t = nT) = 2\cos\frac{2\pi n}{F_s} = 2\cos 0.2\pi n \qquad T = 0.1 \text{ s}$$

Then, $x(n)$ is a periodic sequence with fundamental period $N = 10$.

a. For $N=10$, we obtain $x_N(n)=2\cos 0.4\pi n, 0\le n\le 9$, which contains one period of $x(n)$. The periodic extension of $x_N(n)$ and the magnitude plot of its DFT are shown in the top row of Figure 4.3. For comparison, the DTFT $X_N(e^{j\omega})$ of $x_N(n)$ is also superimposed on the DFT samples. We observe that the DFT has only two nonzero samples, which together constitute the correct frequency of the analog signal $x_c(t)$. The DTFT has a mainlobe and several sidelobes due to the windowing effect. However, the DFT samples the sidelobes at their zero values, as illustrated in the DFT plot. Another explanation for this behavior is that since the samples in $x_N(n)$ for $N=10$ constitute one full period of $\cos 0.4\pi n$, the 10-point periodic extension of $x_N(n)$, shown in the top left graph of Figure 4.3, results in the original sinusoidal sequences $x(n)$. Thus what the DFT "sees" is the exact sampled signal $x_c(t)$. In this case, the choice of N is a desirable one.

b. For $N=15$, we obtain $x_N(n)=2\cos 0.4\pi n,\ 0\le n\le 14$, which contains $1\frac{1}{2}$ periods of $x(n)$. The periodic extension of $x_N(n)$ and the magnitude plot of its DFT are shown in the bottom row of Figure 4.3. Once again for comparison, the DTFT $X_N(e^{j\omega})$ of $x_N(n)$ is superimposed on the DFT samples. In this case, the DFT plot looks markedly different from that for $N=10$ although the DTFT plot appears to be similar. In this case, the DFT does not sample two peaks at the exact frequencies; hence if the resulting DFT samples are joined by the linear interpolation, then we will get a misleading result. Since the sequence $x_N(n)$ does not contain full periods of $\cos 0.4\pi n$, the periodic extension of $x_N(n)$ contains discontinuities at $n=lN$, $l=0, \pm 1, \pm 2, \cdots$, as shown in the bottom left graph of Figure 4.3. This discontinuity results in higher-order harmonics in the DFT values. The DTFT plot also has mainlobes and sidelobes, but the DFT samples these sidelobes at nonzero values. Therefore, the length of the window is an important consideration in spectrum estimation. The sidelobes are the source of the problem of leakage that gives rise to bias in the spectral values, as we will see in the following section. The suppression of the sidelobes is controlled by the window shape, which is another important consideration in spectrum estimation.

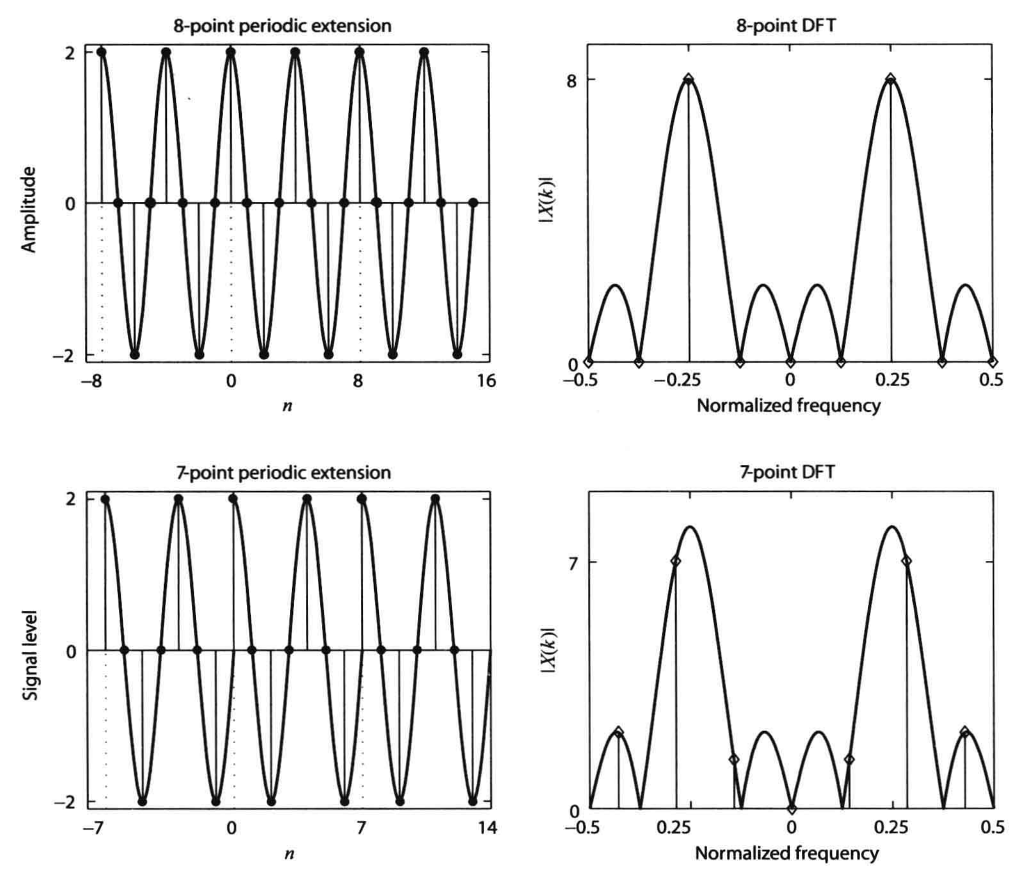


FIGURE 4.3

Effect of window length L on the DFT spectrum shape.

A quantitative description of the above interpretations and arguments related to the capacities and limitations of the DFT is offered by the following result (see Proakis and Manolakis 1996).

THEOREM 4.1 (DFT SAMPLING THEOREM)]. Let $x_c(t)$, $-\infty < t < \infty$, be a continuous-time signal with Fourier transform $X_c(F)$, $-\infty < F < \infty$. Then, the N-point sequences $\{Tx_p(n), 0 \le n \le N-1\}$ and $\{\tilde{X}_p(k), 0 \le k \le N-1\}$ form an N-point DFT pair, that is,

$$x_p(n) \triangleq \sum_{m=-\infty}^{\infty} x_c(nT - mNT) N \overset{\text{DFT}}{\underset{N}{\longleftrightarrow}} \tilde{X}_p(k) \triangleq F_s \sum_{l=-\infty}^{\infty} X_c\left(k\frac{F_s}{N} - lF_s\right) \tag{4.1.6}$$

where $F_s = 1/T$ is the sampling frequency.

Proof. The proof is explored in Problem 4.1.

Thus, given a continuous-time signal $x_c(t)$ and its spectrum $X_c(F)$, we can create a DFT pair by sampling and aliasing in the time and frequency domains. Obviously, this DFT pair provides a "faithful" description of $x_c(t)$ and $X_c(F)$ if both the time-domain aliasing and the frequency-domain aliasing are insignificant. The meaning of relation (4.1.6) is graphically illustrated in Figure 4.4. In this figure, we show the time-domain signals in the left column and their Fourier transforms in the right column. The top row contains continuous-time signals, which are shown as nonperiodic and of infinite extent in both domains, since many real-world signals exhibit this behavior. The middle row contains the sampled version of the continuous-time signal and its periodic Fourier transform (the nonperiodic transform is shown as a dashed curve). Clearly, aliasing in the frequency domain is evident. Finally, the bottom row shows the sampled (periodic) Fourier transform and its correponding time-domain periodic sequence. Again, aliasing in the time domain should be expected. Thus we have sampled and periodic signals in both domains with the certainty of aliasing one domain and the possibility in both domains. This figure should be recalled any time we use the DFT for the analysis of sampled signals.

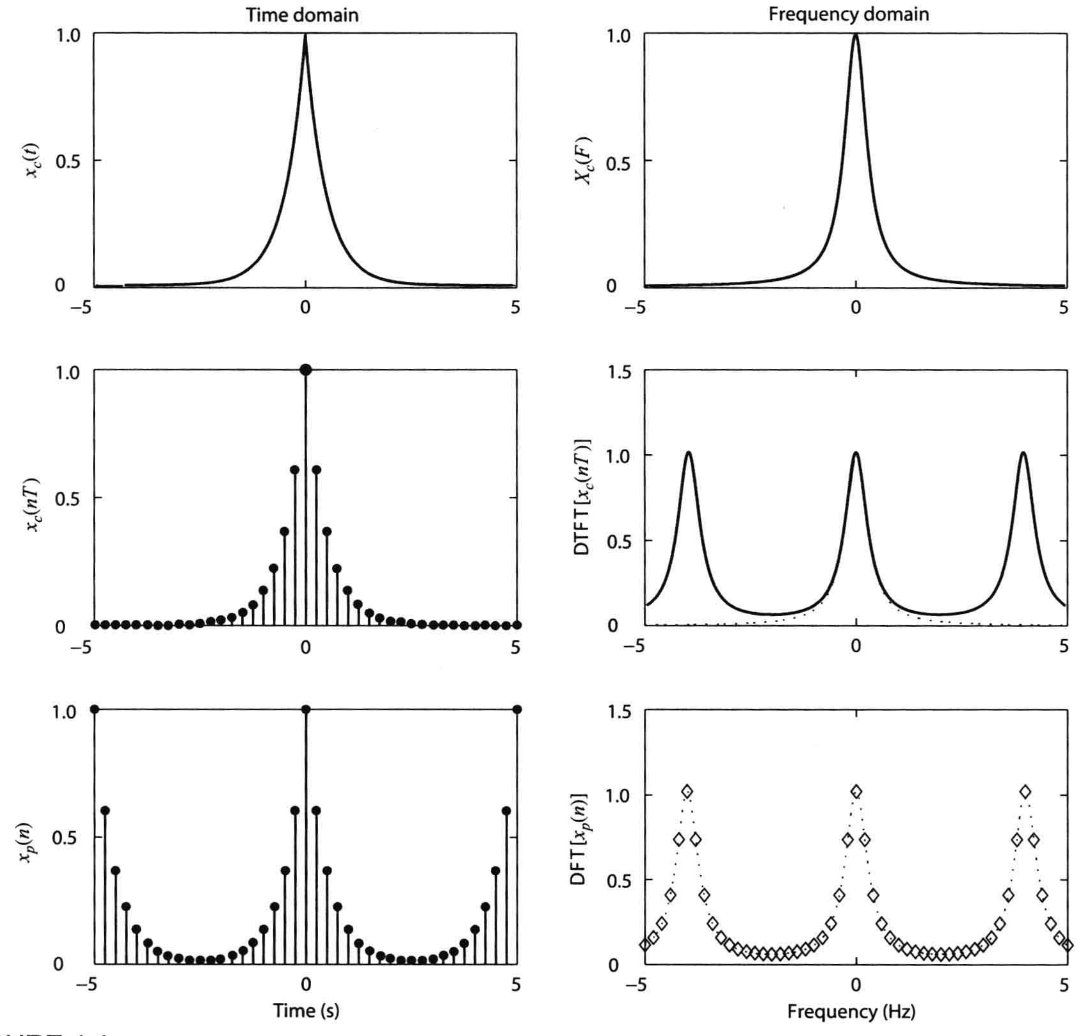


FIGURE 4.4
Graphical illustration of the DFT sampling theorem.

Zero padding

The N-point DFT values of an N-point sequence $x(n)$ are samples of the DTFT $X(e^{j\omega})$. These samples can be used to reconstruct the DTFT $X(e^{j\omega})$ by using the periodic sinc interpolating function. Alternatively, one can obtain more (i.e., dense) samples of the DTFT by computing a larger N_{FFT}-point DFT of $x(n)$, where $N_{FFT} \gg N$. Since the number of samples of $x(n)$ is fixed, the only way we can treat $x(n)$ as an N_{FFT}-point sequence is by appending $N_{FFT} - N$ zeros to it. This procedure is called the *zero padding* operation, and it is used for many purposes including the augmentation of the sequence length so that a power-of-2 FFT algorithm can be used. In spectrum estimation, zero padding is primarily used to provide a *better-looking* plot of the spectrum of a finite-length sequence. This is shown in Figure 4.5 where the magnitude of an N_{FFT}-point DFT of the eight-point sequence $x(n) = \cos(2\pi n/4)$ is plotted for $N_{FFT} = 8$, 16, 32, and 64. The DTFT magnitude $|X(e^{j\omega})|$ is also shown for comparison. It can be seen that as more zeros are appended (by increasing N_{FFT}), the resulting larger-point DFT provides more closely spaced samples of the DTFT, thus giving a better-looking plot. Note, however, that the zero padding *does not increase* the resolution of the spectrum; that is, there are no new peaks and valleys in the display, just a better display of the available information. This type of plot is called a *high-density spectrum*. For a *high-resolution spectrum*, we have to collect more information by increasing N. The DTFT plots shown in Figures 4.3 and 4.5 were obtained by using a very large amount of zero padding.

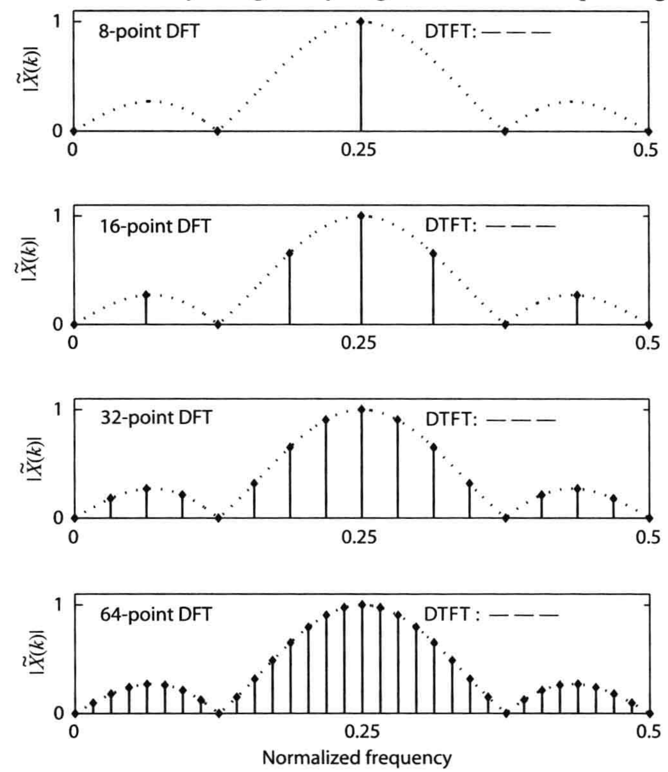


FIGURE 4.5
Effect of zero padding.

4.1.4 Effects of Windowing: Leakage and Loss of Resolution

To see the effect of the window on the spectrum of an arbitrary deterministic signal $x(n)$, defined over the entire range $-\infty < n < \infty$, we notice that the available data record can be expressed as

$$x_N(n) = x(n)w_R(n) \tag{4.1.7}$$

where $w_R(n)$ is the rectangular window defined in (4.1.3). Thus, a finite segment of the signal can be thought of as a product of the actual signal $x(n)$ and a *data window* $w(n)$. In (4.1.7), $w(n) = w_R(n)$, but $w(n)$ can be any arbitrary finite-duration sequence. The Fourier transform of $x_N(n)$ is

$$X_N(e^{j\omega}) = X(e^{j\omega}) \otimes W(e^{j\omega}) \triangleq \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})W(e^{j(\omega-\theta)})\,d\theta \tag{4.1.8}$$

that is, $X_N(e^{j\omega})$ equals the periodic convolution of the actual Fourier transform with the Fourier transform $W(e^{j\omega})$ of the data window. For the rectangular window, $W(e^{j\omega}) = W_R(e^{j\omega})$, where

$$W_R(e^{j\omega}) = \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)}\right] e^{-j\omega(N-1)/2} \triangleq A(\omega)e^{-j\omega(N-1)/2} \tag{4.1.9}$$

The function $A(\omega)$ is a periodic function in ω with fundamental period equal to 2π and is called a *periodic sinc* function. Figure 4.6 shows three periods of $A(\omega)$ for $N=11$. We note that $W_R(e^{j\omega})$ consists of a mainlobe (ML).

$$W_{ML}(e^{j\omega}) = \begin{cases} W_R(e^{j\omega}) & |\omega| < \dfrac{2\pi}{N} \\ 0 & \dfrac{2\pi}{N} < |\omega| \le \pi \end{cases} \tag{4.1.10}$$

and the sidelobes $W_{SL}(e^{j\omega}) = W_R(e^{j\omega}) - W_{ML}(e^{j\omega})$. Thus, (4.1.8) can be written as

$$X_N(e^{j\omega}) = X(e^{j\omega}) \otimes W_{ML}(e^{j\omega}) + X(e^{j\omega}) \otimes W_{SL}(e^{j\omega}) \tag{4.1.11}$$

The first convolution in (4.1.11) smoothes rapid variations and suppresses narrow peaks in $X(e^{j\omega})$, whereas the second convolution introduces ripples in smooth regions of $X(e^{j\omega})$ and can create "false" peaks. Therefore, the spectrum we observe is the convolution of the actual spectrum with the Fourier transform of the data window. The only way to improve the estimate is to increase the window length N or to choose another window shape. For the rectangular window, increasing N results in a narrower mainlobe, and the distortion is reduced. As $N \to \infty, W_R(e^{j\omega})$ tends to an impulse train with period 2π and $X_N(e^{j\omega})$ tends to $X(e^{j\omega})$, as expected. Since in practice the value of N is always finite, the only way to improve the estimate $X_N(e^{j\omega})$ is by properly choosing the shape of the window $w(n)$. The only restriction on $w(n)$ is that it be of finite duration.

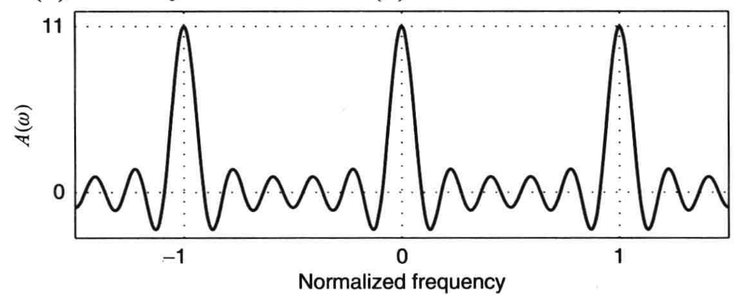


FIGURE 4.6
Plot of $A(\omega) = \sin(\omega N/2)/\sin(\omega/2)$ for $N=11$.

It is known that any time-limited sequence $w(n)$ has a Fourier transform $W(e^{j\omega})$ that is nonzero except at a finite number of frequencies. Thus, from (4.1.8) we see that the estimated value $X_N(e^{j\omega})$ is computed by using all values of $X(e^{j\omega})$ weighted by $W(e^{j(\omega_0-\theta)})$. The contribution of the sinusoidal components with frequencies $\omega \ne \omega_0$ to the value $X_N(e^{j\omega})$ introduces an error known as *leakage*. As the name suggests, energy from one frequency range "leaks" into another, giving the wrong impression of stronger or weaker frequency components.

To illustrate the effect of the window shape and duration on the estimated spectrum, consider the signal

$$x(n) = \cos 0.35\pi n + \cos 0.4\pi n + 0.25\cos 0.8\pi n \tag{4.1.12}$$

which has a line spectrum with lines at frequencies $\omega_1 = 0.35\pi, \omega_2 = 0.4\pi$, and $\omega_3 = 0.8\pi$. This line spectrum (normalized so that the magnitude is between 0 and 1) is shown in the top graph of Figure 4.7 over $0 \le \omega \le \pi$. The spectrum $X_N(e^{j\omega})$ of $x_N(n)$ using the rectangular window is given by

$$\begin{aligned} X_N(e^{j\omega}) = \; & \frac{1}{2}[W(e^{j(\omega+\omega_1)}) + W(e^{j(\omega-\omega_1)}) + W(e^{j(\omega+\omega_2)}) + W(e^{j(\omega-\omega_2)}) \\ & + 0.25W(e^{j(\omega+\omega_3)}) + 0.25W(e^{j(\omega-\omega_3)})] \end{aligned} \tag{4.1.13}$$

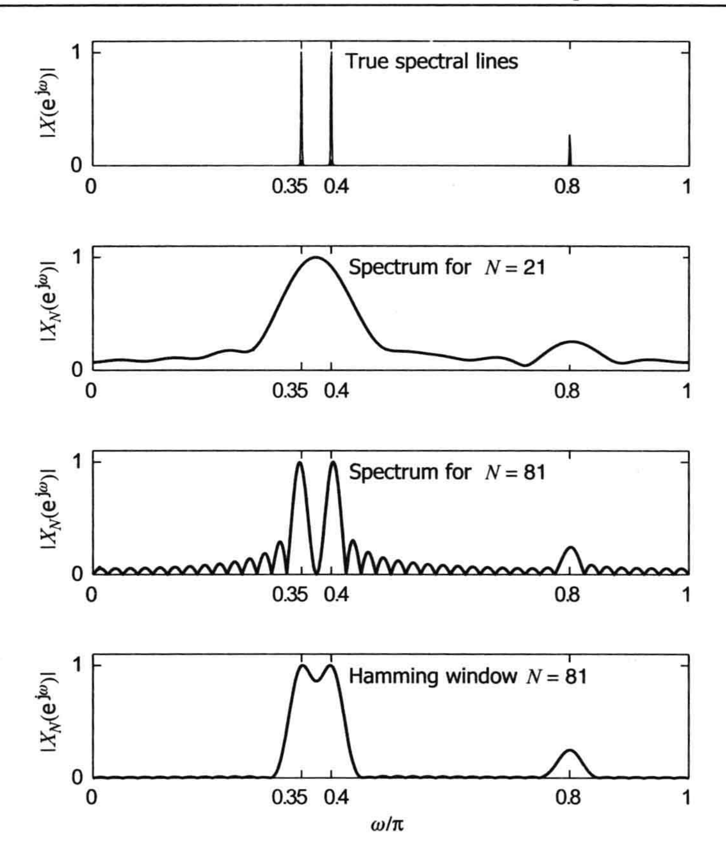


FIGURE 4.7
Spectrum of three sinusoids using rectangular and Hamming windows.

The second and the third plots in Figure 4.7 show 2048-point DFTs of $x_N(n)$ for a rectangular data window with $N = 21$ and $N = 81$. We note that the ability to pick out peaks (*resolvability*) depends on the duration $N-1$ of the data window.[2] To resolve two spectral lines at $\omega = \omega_1$ and $\omega = \omega_2$ using a rectangular window, we should have the difference $|\omega_1 - \omega_2|$ greater than the mainlobe width $\Delta\omega$, which is approximately equal to $2\pi/(N-1)$, in radians per sampling interval, from the plot of $A(\omega)$ in Figure 4.6, that is,

$$|\omega_1 - \omega_2| > \Delta\omega \approx \frac{2\pi}{N-1} \qquad \text{or} \qquad N > \frac{2\pi}{|\omega_1 - \omega_2|} + 1$$

For a rectangular window of length N, the exact value of $\Delta\omega$ is equal to $1.81\pi/(N-1)$. If N is too small, the two peaks at $\omega = 0.35\pi$ and $\omega = 0.4\pi$ are fused into one, as shown in the $N = 21$ plot. When $N = 81$, the corresponding plot shows a resolvable separation; however, the peaks have shifted somewhat from their true locations. This is called *bias*, and it is a direct result of the leakage from sidelobes. In both cases, the peak at $\omega = 0.8\pi$ can be distinguished easily (but also has a bias).

Another important observation is that the sidelobes of the data window introduce false peaks. For a rectangular window, the peak sidelobe level is 13 dB below zero, which is not a good attenuation. Thus these false peaks have values that are comparable to that of the true peak at $\omega = 0.8\pi$, as shown in Figure 4.7. These peaks can be minimized by reducing the amplitudes of the sidelobes. The rectangular window cannot help in this regard because of Gibb's well-known phenomenon associated with it. We need a different window shape. However, any window other than the rectangular window has a wider mainlobe; hence this reduction can be achieved only at the expense of the resolution. To illustrate this, consider the Hamming (Hm) data window, given by

$$w_{\text{Hm}}(n) = \begin{cases} 0.54 - 0.46\cos 2\pi n / N - 1 & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.1.14}$$

---

[2] Since there are N samples in a data window, the number of intervals or durations is $N-1$.

with the approximate width of the mainlobe equal to $8\pi/(N-1)$ and the exact mainlobe width equal to $6.27\pi/(N-1)$. The peak sidelobe level is 43 dB below zero, which is considerably better than that of the rectangular window. The Hamming window is obtained by using the `hamming(N)` function in MATLAB.

The bottom plot in Figure 4.7 shows the 2048-point DFT of the signal $x_N(n)$ for a Hamming window with $N = 81$. Now the peak at $\omega = 0.8\pi$ is more prominent than before, and the sidelobes are almost suppressed. Note also that since the mainlobe width of the Hamming window is wider, the peaks have a wider base—so much so that the first two frequencies are barely recognized. We can correct this problem by choosing a larger window length. This interplay between the shape and the duration of a window function is one of the important issues and, as we will see in Section 4.3, produces similar effects in the spectral analysis of random signals.

Some useful windows

The design of windows for spectral analysis applications has drawn a lot of attention and is examined in detail in Harris (1978). We have already discussed two windows, namely, the rectangular and the Hamming window. Another useful window in spectrum analysis is due to Hann and is mistakenly known as the Hanning window. There are several such windows with varying degrees of tradeoff between resolution (mainlobe width) and leakage (peak sidelobe level). These windows are known as *fixed* windows since each provides a fixed amount of leakage that is independent of the length N. Unlike fixed windows, there are windows that contain a design parameter that can be used to trade between resolution and leakage. Two such windows are the Kaiser window and the Dolph-Chebyshev window, which are widely used in spectrum estimation. Figure 4.8 shows the time-domain window functions and their corresponding frequency-domain log-magnitude plots in decibels for these five windows. The important properties such as peak sidelobe level and mainlobe width of these windows are compared in Table 4.1.

Table 4.1
Comparison of properties of commonly used windows. Each window is assumed to be of length N.

Window type	**Peak sidelobe** level (dB)	**Approximate** mainlobe width	**Exact** mainlode width
Rectangular	-13	$\frac{4\pi}{N-1}$	$\frac{1.81\pi}{N-1}$
Hanning	-32	$\frac{8\pi}{N-1}$	$\frac{5.01\pi}{N-1}$
Hamming	-43	$\frac{8\pi}{N-1}$	$\frac{6.27\pi}{N-1}$
Kaiser	$-A$	—	$\frac{A-8}{2.285N-1}$
Dolph-Chebyshev	$-A$	—	$\cos^{-1}\left[\left(\cosh\frac{\cosh^{-1}10^{A/20}}{N-1}\right)^{-1}\right]$

Hanning window. This window is given by the function

$$w_{\text{Hn}}(n) = \begin{cases} 0.5 - 0.5\cos 2\pi n / N-1 & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.1.15}$$

which is a raised cosine function. The peak sidelobe level is 32 dB below zero, and the approximate mainlobe width is $8\pi/(N-1)$ while the exact mainlobe width is $5.01\pi/(N-1)$. In MATLAB this window function is obtained through the function hanning(N).

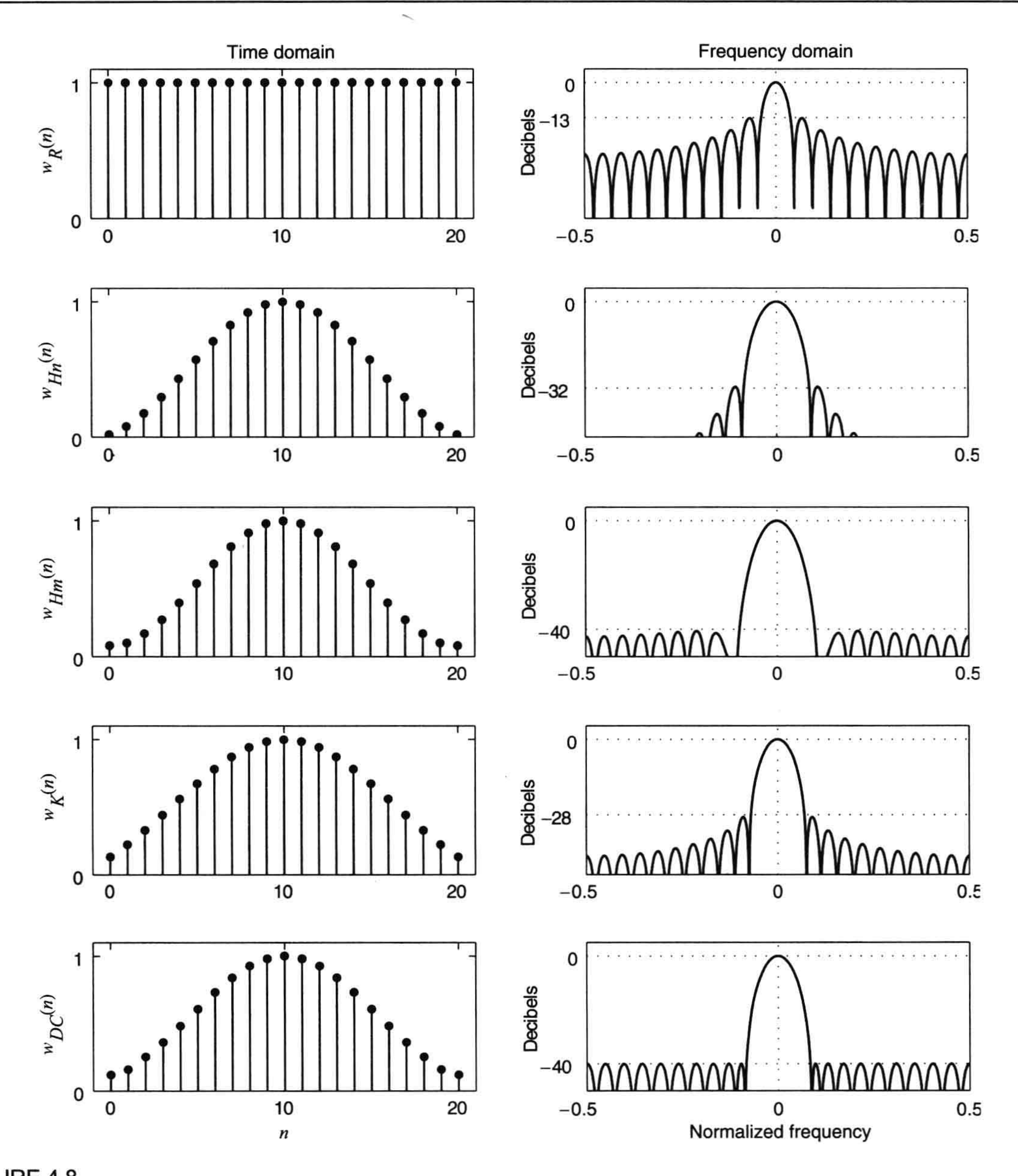


FIGURE 4.8

Time-domain window functions and their frequency-domain characteristics for rectangular, Hanning, Hamming, Kaiser, and Dolph-Chebyshev windows.

Kaiser window. This window function is due to J. F. Kaiser and is given by

$$w_K(n) = \begin{cases} \dfrac{I_0\left\{\beta\sqrt{1-[1-2n/(N-1)]^2}\right\}}{I_0(\beta)} & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.1.16}$$

where $I_0(\cdot)$ is the modified zero-order Bessel function of the first kind and β is a window shape parameter that can be chosen to obtain various peak sidelobe levels and the corresponding mainlobe widths. Clearly, $\beta = 0$ results in the rectangular window while $\beta > 0$ results in lower sidelobe leakage at the expense of a wider mainlobe. Kaiser has developed approximate design equations for β. Given a peak sidelobe level of A dB below the peak value, the approximate value of β is given by

$$\beta = \begin{cases} 0 & A \le 21 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 < A \le 50 \\ 0.1102(A-8.7) & A > 50 \end{cases} \tag{4.1.17}$$

Furthermore, to achieve the given values of the peak sidelobe level of A and the mainlobe width $\Delta\omega$, the length N must satisfy

$$\Delta\omega = \frac{A-8}{2.285(N-1)} \tag{4.1.18}$$

In MATLAB this window is given by the function Kaiser(N, beta).

Dolph-Chebyshev window. This window is characterized by the property that the peak sidelobe levels are constant; that is, it has an "equiripple" behavior. The window $w_{DC}(n)$ is obtained as the inverse DFT of the Chebyshev polynomial evaluated at N equally spaced frequencies around the unit circle. The details of this window function computation are available in Harris (1978). The parameters of the Dolph-Chebyshev window are the constant sidelobe level A in decibels, the window length N, and the mainlobe width $\Delta\omega$. However, only two of the three parameters can be independently specified. In spectrum estimation, parameters N and A are generally specified. Then $\Delta\omega$ is given by

$$\Delta\omega = \cos^{-1}\left[\left(\cosh \frac{\cosh^{-1} 10^{A/20}}{N-1}\right)^{-1}\right] \tag{4.1.19}$$

In MATLAB this window is obtained through the function `chebwin(N,A)`.

To illustrate the usefulness of these windows, consider the same signal containing three frequencies given in (4.1.12). Figure 4.9 shows the spectrum of $x_N(n)$ using the Hanning, Kaiser, and Chebyshev windows for length $N = 81$. The Kaiser and Chebyshev window parameters are adjusted so that the peak sidelobe level is 40 dB or below. Clearly, these windows have suppressed sidelobes considerably compared to that of the rectangular window but the main peaks are wider with negligible bias. The two peaks in the Hanning window spectrum are barely resolved because the mainlobe width of this window is much wider than that of the rectangular window. The Chebyshev window spectrum has uniform sidelobes while the Kaiser window spectrum shows decreasing sidelobes away from the mainlobes.

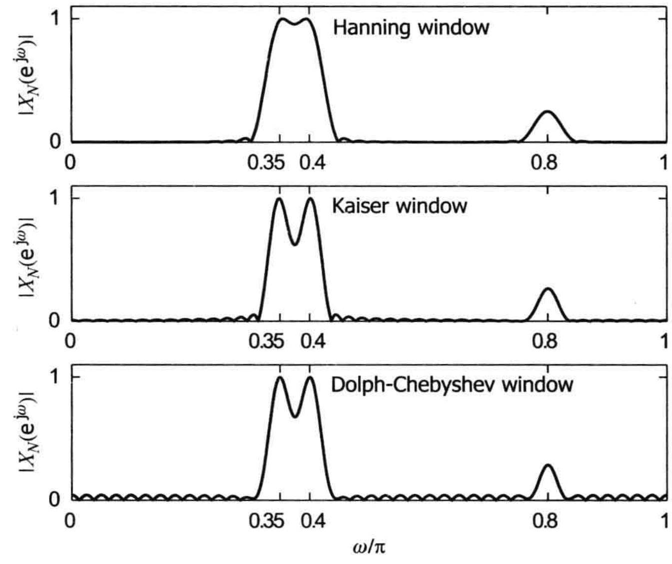


FIGURE 4.9
Spectrum of three sinusoids using Hanning, Kaiser, and Chebyshev windows.

4.1.5 Summary

In conclusion, the frequency analysis of deterministic signals requires a careful study of three important steps. First,

the continuous-time signal $x_c(t)$ is sampled to obtain samples $x(n)$ that are collected into blocks or frames. The frames are "conditioned" to minimize certain errors by multiplying by a window sequence $w(n)$ of length N. Finally the windowed frames $x_N(n)$ are transformed to the frequency domain using the DFT. The resulting DFT spectrum $\tilde{X}_N(k)$ is a *faithful* replica of the actual spectrum $X_c(F)$ if the following errors are sufficiently small.

Aliasing error. This is an error due to the sampling operation. If the sampling rate is sufficiently high and if the antialiasing filter is properly designed so that most of the frequencies of interest are represented in $x(n)$, then this error can be made smaller. However, a certain amount of aliasing should be expected.

Errors due to finite-length window. There are several errors such as resolution loss, bias, and leakage that are attributed to the windowing operation. Therefore, a careful design of the window function and its length is necessary to minimize these errors. These topics were discussed in Section 4.1.4. In Table 4.1 we summarize key properties of five windows discussed in this section that are useful for spectrum estimation.

Spectrum reconstruction error. The DFT spectrum $\tilde{X}_N(k)$ is a number sequence that must be reconstructed into a continuous function for the purpose of plotting. A practical choice for this reconstruction is the first-order polynomial interpolation. This reconstruction error can be made smaller (and in fact comparable to the screen resolution) by choosing a large number of frequency samples, which can be achieved by the *zero padding* operation in the DFT. It was discussed in Section 4.1.3.

With the understanding of frequency analysis concepts developed in this section, we are now ready to tackle the problem of spectral analysis of stationary random signals. From Chapter 2 we recognize that the true spectral values can only be obtained as estimates. This requires some understanding of key concepts from estimation theory, which is developed in Section 2.4.

4.2 Estimation of the Autocorrelation of Stationary Random Signals

The second-order moments of a stationary random sequence—that is, the mean value μ_x, the autocorrelation sequence $r_x(l)$, and the PSD $R_x(e^{j\omega})$—play a crucial role in signal analysis and signal modeling. In this section, we discuss the estimation of the autocorrelation sequence $r_x(l)$ using a finite data record $\{x(n)\}_0^{N-1}$ of the process.

For a stationary process $x(n)$, the most widely used estimator of $r_x(l)$ is given by the *sample autocorrelation sequence*

$$\hat{r}_x(l) \triangleq \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \le l \le N-1 \\ \hat{r}_x^*(-l) & -(N-1) \le l < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.1}$$

or, equivalently,

$$\hat{r}_x(l) \triangleq \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=l}^{N-1} x(n)x^*(n-l) & 0 \le l \le N-1 \\ \hat{r}_x^*(-l) & -(N-1) \le l < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.2}$$

which is a random sequence. Note that without further information beyond the observed data $\{x(n)\}_0^{N-1}$, it is not possible to provide reasonable estimates of $r_x(l)$ for $|l| \ge N$. Even for lag values $|l|$ close to N, the correlation estimates are unreliable since very few $x(n+|l|)x(n)$ pairs are used. A good rule of thumb provided by Box and Jenkins (1976) is that N should be at least 50 and that $|l| \le N/4$. The sample autocorrelation $\hat{r}_x(l)$ given in (4.2.1) has a desirable property that for each $l \ge 1$, the sample autocorrelation matrix

$$\hat{\mathbf{R}}_x = \begin{bmatrix} \hat{r}_x(0) & \hat{r}_x^*(1) & \cdots & \hat{r}_x^*(N-1) \\ \hat{r}_x(1) & \hat{r}_x(0) & \cdots & \hat{r}_x^*(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_x(N-1) & \hat{r}_x(N-2) & \cdots & \hat{r}_x(0) \end{bmatrix} \tag{4.2.3}$$

is nonnegative definite (see Section 2.3.1). This property is explored in Problem 4.5. MATLAB provides functions to compute the correlation matrix $\hat{\mathbf{R}}_x$ (for example, `corr`), given the data $\{x(n)\}_{n=0}^{N-1}$; however, the book toolbox function `rx = autoc(x,L);` computes $\hat{r}_x(l)$ according to (4.2.1) very efficiently.

The estimate of covariance $\gamma_x(l)$ from the data record $\{x(n)\}_0^{N-1}$ is given by the sample autocovariance sequence

$$\hat{\gamma}_x(l) = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-l-1}[x(n+l)-\hat{\mu}_x][x^*(n)-\hat{\mu}_x^*] & 0 \le l \le N-1 \\ \hat{\gamma}_x^*(-l) & -(N-1) \le l < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.4}$$

so that the corresponding autocovariance matrix $\hat{\Gamma}_x$ is nonnegative definite. Similarly, the sample autocorrelation coefficient sequence $\hat{\rho}_x(l)$ is given by

$$\hat{\rho}_x(l) = \frac{\hat{\gamma}_x(l)}{\hat{\sigma}_x^2} \tag{4.2.5}$$

In the rest of this section, we assume that $x(n)$ is a zero-mean process and hence $\hat{r}_x(l) = \hat{\gamma}_x(l)$, so that we can discuss the autocorrelation estimate in detail.

To determine the statistical quality of this estimator, we now consider its mean and variance.

Mean of $\hat{r}_x(l)$. We first note that (4.2.1) can be written as

$$\hat{r}_x(l) = \frac{1}{N}\sum_{n=-\infty}^{\infty} x(n+l)w(n+l)x^*(n)w(n) \qquad |l| \ge 0 \tag{4.2.6}$$

where

$$w(n) = w_R(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.7}$$

is the rectangular window. The expected value of $\hat{r}_x(l)$ is

$$E\{\hat{r}_x(l)\} = \frac{1}{N}\sum_{n=-\infty}^{\infty} E\{x(n+l)x^*(n)\}w(n+l)w(n) \qquad l \ge 0$$

and

$$E\{\hat{r}_x(-l)\} = E\{\hat{r}_x^*(l)\} \qquad -l \le 0$$

Therefore

$$E\{\hat{r}_x(l)\} = \frac{1}{N}r_x(l)r_w(l) \tag{4.2.8}$$

where

$$r_w(l) = w(l) * w(-l) = \sum_{n=-\infty}^{\infty} w(n)w(n+l) \tag{4.2.9}$$

is the autocorrelation of the window sequence. For the rectangular window

$$r_w(l) = w_B(n) \triangleq \begin{cases} N-|l| & |l| \le N-1 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.10}$$

which is the unnormalized triangular or Bartlett windows. Thus

$$E\{\hat{r}_x(l)\} = \frac{1}{N}r_x(l)w_B(n) = r_x(l)\left(1-\frac{|l|}{N}\right)w_R(n) \tag{4.2.11}$$

Therefore, we conclude that the relation (4.2.1) provides a *biased* estimate of $r_x(l)$ because the expected value of $\hat{r}_x(l)$ from (4.2.11) is not equal to the true autocorrelation $r_x(l)$. However, $\hat{r}_x(l)$ is an *asymptotically unbiased* estimator since if $N \to \infty, E\{\hat{r}_x(l)\} \to r_x(l)$. Clearly, the bias is small if $\hat{r}_x(l)$ is evaluated for $|l| \le L$, where L

is the maximum desired lag and $L \ll N$.

Variance of $\hat{r}_x(l)$. An approximate expression for the covariance of $\hat{r}_x(l)$ is given by Jenkins and Watts (1968)

$$\text{cov}\{\hat{r}_x(l_1), \hat{r}_x(l_2)\} \simeq \frac{1}{N} \sum_{l=-\infty}^{\infty} [r_x(l) r_x(l+l_2-l_1) + r_x(l+l_2) r_x(l-l_1)] \tag{4.2.12}$$

This indicates that successive values of $\hat{r}_x(l)$ may be highly correlated and that $\hat{r}_x(l)$ may fail to die out even if it is expected to. This makes the interpretation of autocorrelation graphs quite challenging because we do not know whether the variation is real or statistical.

The variance of $\hat{r}_x(l)$, which can be obtained by setting $l_1 = l_2$ in (4.2.12), tends to zero as $N \to \infty$. Thus, $\hat{r}_x(l)$ provides a good estimate of $r_x(l)$ if the lag $|l|$ is much smaller than N. However, as $|l|$ approaches N, fewer and fewer samples of $x(n)$ are used to evaluate $\hat{r}_x(l)$. As a result, the estimate $\hat{r}_x(l)$ becomes worse and its variance increases.

Nonnegative definiteness of $\hat{r}_x(l)$. An alternative estimator for the autocorrelation sequence is given by

$$\check{r}_x(l) = \begin{cases} \dfrac{1}{N-l} \displaystyle\sum_{n=0}^{N-l-1} x(n+l) x^*(n) & 0 \le l \le L < N \\ \check{r}_x^*(-l) & -N < -L \le l < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4.2.13}$$

Although this estimator is unbiased, it is not used in spectral estimation because of its negative definiteness. In contrast, the estimator $\hat{r}_x(l)$ from (2.4.33) is nonnegative definite, and any spectral estimates based on it do not have any negative values. Furthermore, the estimator $\hat{r}_x(l)$ has smaller variance and mean square error than the estimator $\check{r}_x(l)$ (Jenkins and Watts 1968). Thus, in this book we use the estimator $\hat{r}_x(l)$ defined in (4.2.1).

4.3 Estimation of the Power Spectrum of Stationary Random Signals

From a practical point of view, most stationary random processes have continuous spectra. However, harmonic processes (i.e., processes with line spectra) appear in several applications either alone or in mixed spectra (a mixture of continuous and line spectra). We first discuss the estimation of continuous spectra in detail. The estimation of line spectra is considered in Chapter 8.

The power spectral density of a zero-mean stationary stochastic process was defined in (2.1.44) as

$$R_x(e^{j\omega}) \triangleq \sum_{l=-\infty}^{\infty} r_x(l) e^{-j\omega l} \tag{4.3.1}$$

assuming that the autocorrelation sequence $r_x(l)$ is absolutely summable. We will deal with the problem of estimating the power spectrum $R_x(e^{j\omega})$ of a stationary process $x(n)$ from a finite record of observations $\{x(n)\}_0^{N-1}$ of a single realization. The ideal goal is to devise an estimate that will faithfully characterize the power-versus-frequency distribution of the stochastic process (i.e., all the sequences of the ensemble) using only a segment of a single realization. For this to be possible, the estimate should typically involve some kind of averaging among several realizations or along a single realization.

In some practical applications (e.g., interferometry), it is possible to directly measure the autocorrelation $r_x(l)$, $|l| \le L < N$ with great accuracy. In this case, the spectrum estimation problem can be treated as a deterministic one, as described in Section 4.1. We will focus on the "stochastic" version of the problem, where $R_x(e^{j\omega})$ is estimated from the available data $\{x(n)\}_0^{N-1}$. A natural estimate of $R_x(e^{j\omega})$, suggested by (4.3.1), is to estimate $r_x(l)$ from the available data and then transform it by using (4.3.1).

4.3.1 Power Spectrum Estimation Using the Periodogram

The periodogram is an estimator of the power spectrum, introduced by Schuster (1898) in his efforts to search for hidden periodicities in solar sunspot data. The *periodogram* of the data segment $\{x(n)\}_0^{N-1}$ is defined by

$$\hat{R}_x(e^{j\omega}) \triangleq \frac{1}{N}\left|\sum_{n=0}^{N-1} v(n)e^{-j\omega n}\right|^2 = \frac{1}{N}|V(e^{j\omega})|^2 \tag{4.3.2}$$

where $V(e^{j\omega})$ is the DTFT of the windowed sequence

$$v(n) = x(n)w(n) \qquad 0 \le n \le N-1 \tag{4.3.3}$$

The above definition of the periodogram stems from Parseval's relation on the power of a signal. The window $w(n)$, which has length N, is known as the *data window*. Usually, the term *periodogram* is used when $w(n)$ is a rectangular window. In contrast, the term *modified periodogram* is used to stress the use of nonrectangular windows. The values of the periodogram at the discrete set of frequencies $\{\omega_k = 2\pi k/N\}_0^{N-1}$ can be calculated by

$$\tilde{\hat{R}}_x(k) \triangleq \hat{R}_x(e^{j2\pi k/N}) = \frac{1}{N}|\tilde{V}(k)|^2 \qquad k = 0,\ 1,\ \cdots,\ N-1 \tag{4.3.4}$$

where $\tilde{V}(k)$ is the N-point DFT of the windowed segment $v(n)$. In MATLAB, the modified periodogram computation is implemented by using the function

```
Rx = psd(x, Nfft, Fs, window(N), 'none');
```

where `window` is the name of any MATLAB-provided window function (e.g., `hamming`); `Nfft` is the size of the DFT, which is chosen to be larger than N to obtain a high-density spectrum (see zero padding in Section 4.1.1); and `Fs` is the sampling frequency, which is used for plotting purposes. If the window `boxcar` is used, then we obtain the periodogram estimate.

The periodogram can be expressed in terms of the autocorrelation estimate $\hat{r}_v(l)$ of the windowed sequence $v(n)$ as (see Problem 4.9)

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-(N-1)}^{N-1} \hat{r}_v(l)e^{-j\omega l} \tag{4.3.5}$$

which shows that $\hat{R}_x(e^{j\omega})$ is a "natural" estimate of the power spectrum. From (4.3.2) it follows that $\hat{R}_x(e^{j\omega})$ is nonnegative for all frequencies ω. This results from the fact that the autocorrelation sequence $\hat{r}(l)$, $0 \le |l| \le N-1$, is nonnegative definite. If we use the estimate $\check{r}_x(l)$ from (4.2.13) in (4.3.5) instead of $\hat{r}_x(l)$, the obtained periodogram may assume negative values, which implies that $\check{r}_x(l)$ is not guaranteed to be nonnegative definite.

The inverse Fourier transform of $\hat{R}_x(e^{j\omega})$ provides the estimated autocorrelation $\hat{r}_v(l)$, that is,

$$\hat{r}_v(l) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{R}_x(e^{j\omega})e^{j\omega l}\,d\omega \tag{4.3.6}$$

because $\hat{r}_v(l)$ and $\hat{R}_x(e^{j\omega})$ form a DTFT pair. Using (4.3.6) and (4.2.1) for $l = 0$, we have

$$\hat{r}_v(0) = \frac{1}{N}\sum_{n=0}^{N-1}|v(n)|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{R}_x(e^{j\omega})\,d\omega \tag{4.3.7}$$

Thus, the periodogram $\hat{R}_x(e^{j\omega})$ shows how the power of the segment $\{v(n)\}_0^{N-1}$, which provides an estimate of the variance of the process $x(n)$, is distributed as a function of frequency.

Filter bank interpretation. The above assertion that the periodogram describes a distribution of power as a function of frequency can be interpreted in a different way, in which the power estimate over a narrow frequency band is attributed to the output power of a narrow-bandpass filter. This leads to the well-known *filter bank interpretation* of the periodogram. To develop this interpretation, consider the basic (unwindowed) periodogram estimator $\hat{R}_x(e^{j\omega})$ in (4.3.2), evaluated at a frequency $\omega_k \triangleq k\Delta\omega \triangleq 2\pi k/N$, which can be expressed as

$$\begin{aligned}\hat{R}_x(e^{j\omega_k}) &= \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-j\omega_k n}\right|^2 = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{j2\pi k - j\omega_k n}\right|^2 \\ &= \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{j\omega_k (N-n)}\right|^2 \qquad \text{since } \omega_k N = 2\pi k \\ &= \frac{1}{N}\left|\sum_{m=0}^{N-1} x(N-1-m)e^{j\omega_k m}\right|^2\end{aligned} \tag{4.3.8}$$

Clearly, the term inside the absolute value sign in (4.3.8) can be interpreted as a convolution of $x(n)$ and $e^{j\omega_k n}$,

evaluated at $n = N-1$. Define

$$h_k(n) \triangleq \begin{cases} \frac{1}{N}\mathrm{e}^{\mathrm{j}\omega_k n} & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.3.9}$$

as the impulse response of a linear system whose frequency response is given by

$$\begin{aligned} H_k(\mathrm{e}^{\mathrm{j}\omega}) &= \mathcal{F}[h_k(n)] = \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{e}^{\mathrm{j}\omega_k n}\mathrm{e}^{-\mathrm{j}\omega n} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{e}^{-\mathrm{j}(\omega-\omega_k)n} = \frac{1}{N}\frac{\mathrm{e}^{-\mathrm{j}N(\omega-\omega_k)}-1}{\mathrm{e}^{-\mathrm{j}(\omega-\omega_k)}-1} \\ &= \frac{1}{N}\frac{\sin[N(\omega-\omega_k)/2]}{\sin[(\omega-\omega_k)/2]}\mathrm{e}^{-\mathrm{j}(N-1)(\omega-\omega_k)/2} \end{aligned} \tag{4.3.10}$$

which is a linear-phase, narrow-bandpass filter centered at $\omega = \omega_k$. The 3-dB bandwidth of this filter is proportional to $2\pi / N$ rad per sampling interval (or $1/N$ cycles per sampling interval). A plot of the magnitude response $|H_k(\mathrm{e}^{\mathrm{j}\omega})|$, for $\omega_k = \pi/2$ and $N = 50$, is shown in Figure 4.10, which evidently shows the narrowband nature of the filter.

Continuing, we also define the output of the filter $h_k(n)$ by $y_k(n)$, that is,

$$y_k(n) \triangleq h_k(n) * x(n) = \frac{1}{N}\sum_{m=0}^{N-1} x(n-m)\mathrm{e}^{\mathrm{j}\omega_k m} \tag{4.3.11}$$

Then (4.3.8) can be written as

$$\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega_k}) = N\,|\,y_k(N-1)\,|^2 \tag{4.3.12}$$

Now consider the average power in $y_k(n)$, which can be evaluated using the spectral density as

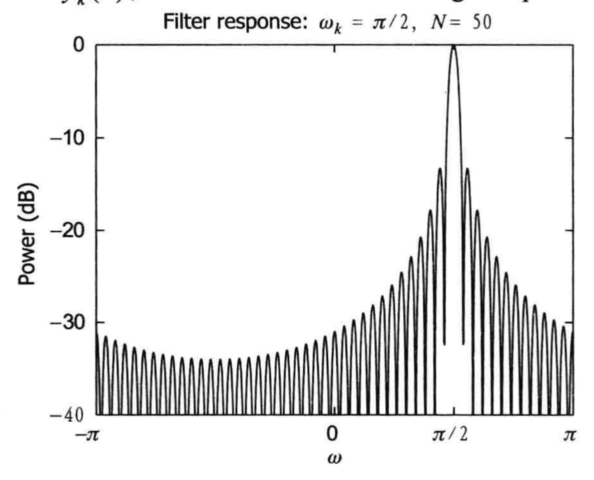


FIGURE 4.10
The magnitude of the frequency response of the narrow-bandpass filter for $\omega_k = \pi/2$ and $N = 50$

$$\begin{aligned} E\{|\,y_k(n)\,|^2\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} R_x(\mathrm{e}^{\mathrm{j}\omega})\,|\,H_k(\mathrm{e}^{\mathrm{j}\omega})\,|^2\,\mathrm{d}\omega \\ &\approx \frac{\Delta\omega}{2\pi}R_x(\mathrm{e}^{\mathrm{j}\omega_k}) = \frac{1}{N}R_x(\mathrm{e}^{\mathrm{j}\omega_k}) \end{aligned} \tag{4.3.13}$$

since $H(e^{j\omega})$ is a narrowband filter. If we estimate the average power $E\{|\,y_k(n)\,|^2\}$ using one sample $y_k(N-1)$, then from (4.3.13) the estimated spectral density is the periodogram given by (4.3.12), which says that the kth DFT sample of the periodogram [see (4.3.4)] is given by the average power of a *single* $(N-1)$*st* output sample of the ω_k-centered narrow-bandpass filter. Now imagine one such filter for each ω_k, $k = 0,\cdots,N-1$, frequencies. Thus we have a bank of filters, each tuned to the discrete frequency (based on the data record length), providing the periodogram estimates every N samples. This filter bank is inherently built into the periodogram and hence need

not be explicitly implemented. The block diagram of this filter bank approach to the periodogram computation is shown in Figure 4.11.

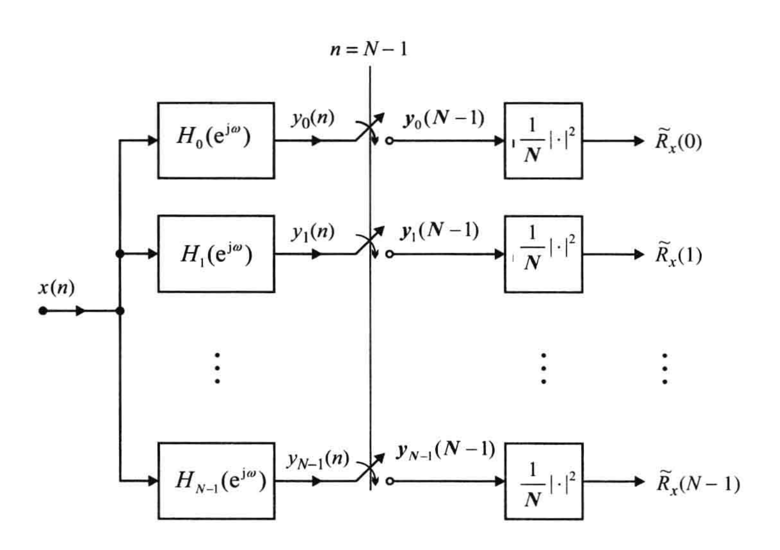


FIGURE 4.11
The filter bank approach to the periodogram computation.

In Section 5.1, we observed that the periodogram of a deterministic signal approaches the true energy spectrum as the number of observations $N \to \infty$. To see how the power spectrum of random signals is related to the number observations, we consider the following example.

EXAMPLE 4.3.1 (PERIODOGRAM OF A SIMULATED WHITE NOISE SEQUENCE). Let $x(n)$ be a stationary white Gaussian noise with zero-mean and unit variance. The theoretical spectrum of $x(n)$ is

$$R_x(e^{j\omega}) = \sigma_x^2 = 1 \qquad -\pi < \omega \leq \pi$$

To study the periodogram estimate, 50 different N-point records of $x(n)$ were generated using a pseudorandom number generator. The periodogram $\hat{R}_x(e^{j\omega})$ of each record was computed for $\omega = \omega_k = 2\pi k/1024$, $k = 0, 1, \cdots, 512$, that is, with $N_{FFT} = 1024$, from the available data using (4.3.4) for $N = 32$, 128, and 256. These results in the form of periodogram overlays (a Monte Carlo simulation) and their averages are shown in Figure 4.12. We notice that $\hat{R}_x(e^{j\omega})$ fluctuates so erratically that it is impossible to conclude from its observation that the signal has a flat spectrum. Furthermore, the size of the fluctuations (as seen from the ensemble average) is not reduced by increasing the segment length N. In this sense, we should not expect the periodogram $\hat{R}_x(e^{j\omega})$ to converge to the true spectrum $R_x(e^{j\omega})$ in some statistical sense as $N \to \infty$. Since $R_x(e^{j\omega})$ is constant over frequency, the fluctuations of $\hat{R}_x(e^{j\omega})$ can be characterized by their mean, variance, and mean square error over frequency for each N and are given in Table 4.2. It can be seen that although the mean value tends to 1 (true value), the standard deviation is not reduced as N increases. In fact, it is close to 1; that is, it is of the order of the size of the quantity to be estimated. This illustrates that the periodogram is not a good estimate of the power spectrum.

Since for each value of ω, $\hat{R}_x(e^{j\omega})$ is a random variable, the erratic behavior of the periodogram estimator, which is illustrated in Figure 4.12, can be explained by considering its mean, covariance, and variance.

Table 4.2
Performance of periodogram for white Gaussian noise signal in Example 43.

N	32	128	256
$E[R_x(e^{j\omega_k})]$	0.7829	0.8954	0.9963
$var[R_x(e^{j\omega_k})]$	0.7232	1.0635	1.1762
MSE	0.7689	1.07244	1.1739

Mean of $\hat{R}_x(e^{j\omega})$. Taking the mathematical expectation of (4.3.5) and using (4.2.8), we obtain

$$E\{\hat{R}_x(e^{j\omega})\} = \sum_{l=-(N-1)}^{N-1} E\{\hat{r}_x(l)\}e^{-j\omega l} = \frac{1}{N}\sum_{l=-(N-1)}^{N-1} r_x(l)r_w(l)e^{-j\omega l} \tag{4.3.14}$$

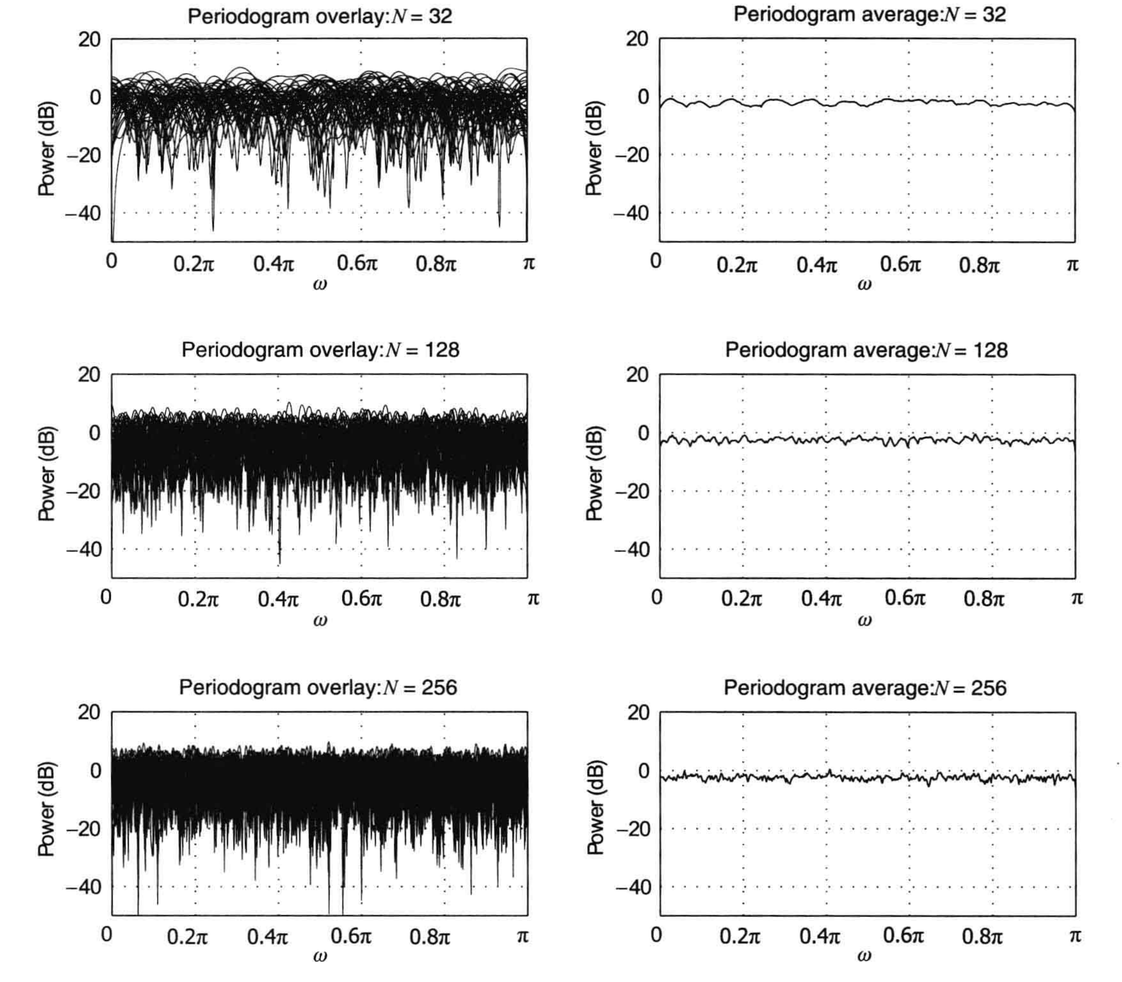


FIGURE 4.12
Periodograms of white Gaussian noise in Example 4.3.1.

Since $E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\} \neq R_x(\mathrm{e}^{\mathrm{j}\omega})$, the periodogram is a biased estimate of the true power spectrum $R_x(\mathrm{e}^{\mathrm{j}\omega})$.

Equation (4.3.14) can be interpreted in the frequency domain as a periodic convolution. Indeed, using the frequency domain convolution theorem, we have

$$E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\} = \frac{1}{2\pi N}\int_{-\pi}^{\pi} R_x(\mathrm{e}^{\mathrm{j}\theta})R_w(\mathrm{e}^{\mathrm{j}(\omega-\theta)})\,\mathrm{d}\theta \tag{4.3.15}$$

where

$$R_w(\mathrm{e}^{\mathrm{j}\omega}) = |W(\mathrm{e}^{\mathrm{j}\omega})|^2 \tag{4.3.16}$$

is the spectrum of the window. Thus, the expected value of the peroodogram is obtained by convolving the true spectrum $R_x(\mathrm{e}^{\mathrm{j}\omega})$ with the spectrum $R_w(\mathrm{e}^{\mathrm{j}\omega})$ of the window. This is equivalent to windowing the true autocorrelation $r_x(l)$ with the *correlation* or *lag* window $r_w(l) = w(l) * w(-l)$, where $w(n)$ is the data window.

To understand the implications of (4.3.15), consider the rectangular data window (4.2.7). Using (4.2.11), we see that (4.3.14) becomes

$$E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{l=-(N-1)}^{N-1}\left(1-\frac{|l|}{N}\right)r_x(l)\mathrm{e}^{-\mathrm{j}\omega l} \tag{4.3.17}$$

For nonperiodic autocorrelations, the value of $r_x(l)$ becomes negligible for large values of $|l|$. Hence, as the record length N increases, the term $(1-|l|/N) \to 1$ for all l, which implies that

$$\lim_{N\to\infty} E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\} = R_x(\mathrm{e}^{\mathrm{j}\omega}) \tag{4.3.18}$$

that is, the periodogram is an *asymptotically unbiased* estimator of $R_x(e^{j\omega})$. In the frequency domain, we obtain

$$R_w(e^{j\omega}) = \mathcal{F}\{w_R(l) * w_R(-l)\} = |W_R(e^{j\omega})|^2 = \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)}\right]^2 \tag{4.3.19}$$

where

$$W_R(e^{j\omega}) = e^{-j\omega(N-1)/2}\frac{\sin(\omega N/2)}{\sin(\omega/2)} \tag{4.3.20}$$

is the Fourier transform of the rectangular window. The spectrum $R_w(e^{j\omega})$, in (4.3.19), of the correlation window $r_w(l)$ approaches a periodic impulse train as the window length increases.[3] As a result, $E\{\hat{R}_x(e^{j\omega})\}$ approaches the true power spectrum $R_x(e^{j\omega})$ as N approaches ∞.

The result (4.3.18) holds for any window that satisfies the following two conditions:

1. The window is normalized such that

$$\sum_{n=0}^{N-1} |w(n)|^2 = N \tag{4.3.21}$$

   This condition is obtained by noting that, for asymptotic unbiasedness, we want $R_w(e^{j\omega})/N$ in (4.3.15) to be an approximation of an impulse in the frequency domain. Since the area under the impulse function is unity, using (4.3.16) and Parseval's theorem, we have

$$\frac{1}{2\pi N}\int_{-\pi}^{\pi} |W(e^{j\omega})|^2\, d\omega = \frac{1}{N}\sum_{n=0}^{N-1} |w(n)|^2 = 1 \tag{4.3.22}$$

2. The width of the mainlobe of the spectrum $R_w(e^{j\omega})$ of the correlation window decreases as $1/N$. This condition guarantees that the area under $R_w(e^{j\omega})$ is concentrated at the origin as N becomes large.

The bias is introduced by the sidelobes of the correlation window through leakage as illustrated in section 4.1. Therefore, we can reduce the bias by using the modified periodogram and a "better" window. Bias can be avoided if either $N = \infty$, in which case the spectrum of the window is a periodic train of impulses, or $R_x(e^{j\omega}) = \sigma_x^2$, that is, $x(n)$ has a flat power spectrum. Thus, for white noise, $\hat{R}_x(e^{j\omega})$ is unbiased for all N. This fact was apparent in Example 4.3.1 and is very important for practical applications. In the following example, we illustrate that the bias becomes worse as the dynamic range of the spectrum increases.

EXAMPLE 4.3.2 (BIAS AND LEAKAGE PROPERTIES OF THE PERIODOGRAM)]. Consider an AR(2) process with

$$\mathbf{a}_2 = [1\ \ -0.75\ \ 0.5]^T \qquad d_0 = 1 \tag{4.3.23}$$

and an AR(4) process with

$$\mathbf{a}_4 = [1\ \ -2.7607\ \ 3.8106\ \ -2.6535\ \ 0.9238]^T \qquad d_0 = 1 \tag{4.3.24}$$

where $w(n) \sim \mathrm{WN}(0,1)$. Both processes have been used extensively in the literature for power spectrum estimation studies (Percival and Walden 1993). Their power spectrum is given by (see Chapter 3)

$$R_x(e^{j\omega}) = \frac{\sigma_w^2 d_0}{|A(e^{j\omega})|^2} = \frac{\sigma_\omega^2}{\left|\sum_{k=0}^{p} a_k e^{j\omega k}\right|^2} \tag{4.3.25}$$

For simulation purposes, $N = 1024$ samples of each process were generated. The sample realizations and the shapes of the two power spectra in (4.3.25) are shown in Figure 4.13. The dynamic range of the two spectra, that is, $\max_\omega R_x(e^{j\omega})/\min_\omega R_x(e^{j\omega})$, is about 15 and 65 dB, respectively.

From the sample realizations, periodograms and modified periodograms, based on the Hanning window, were computed by using (4.3.4) at $N_{\mathrm{FFT}} = 1024$ frequencies. These are shown in Figure 4.14. The periodograms for the AR(2) and AR(4) processes, respectively, are shown in the top row while the modified periodograms for the same processes are shown in the bottom row. These plots illustrate that the periodogram is a biased estimator of the power spectrum. In the case of the AR(2) process, since the spectrum has a small dynamic range (15 dB), the bias in the periodogram estimate is not obvious; furthermore, the windowing in the modified periodogram did not show much improvement. On the other hand, the AR(4) spectrum has a large dynamic range, and hence the bias is clearly visible at high frequencies. This bias is clearly reduced by windowing of the data in the modified periodogram. In both cases, the random fluctuations are not reduced by the data windowing operation.

---

[3]This spectrum is sometimes referred to as the *Fejer kernel.*

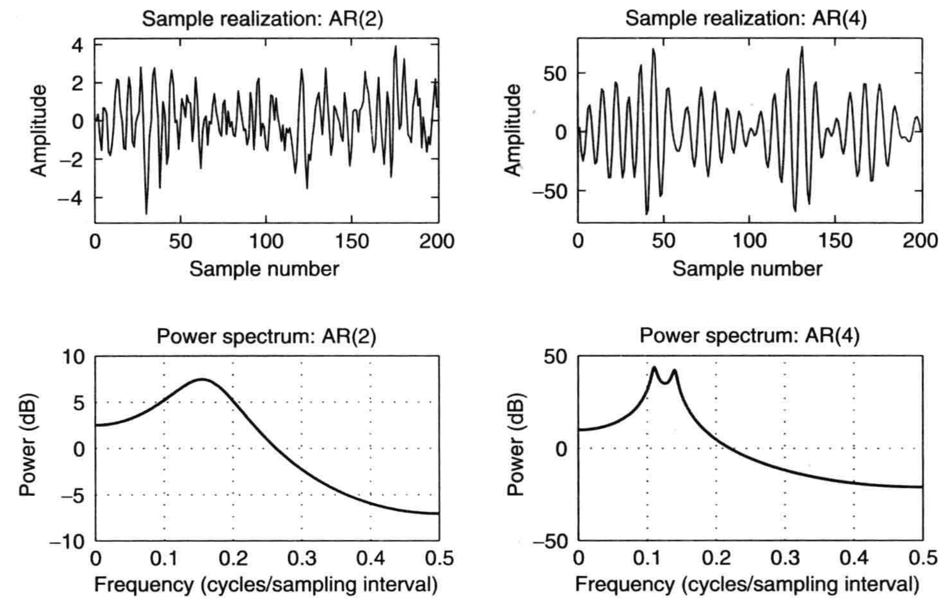


FIGURE 4.13
Sample realizations and power spectra of the AR(2) and AR(4) processes used in Example 4.3.2.

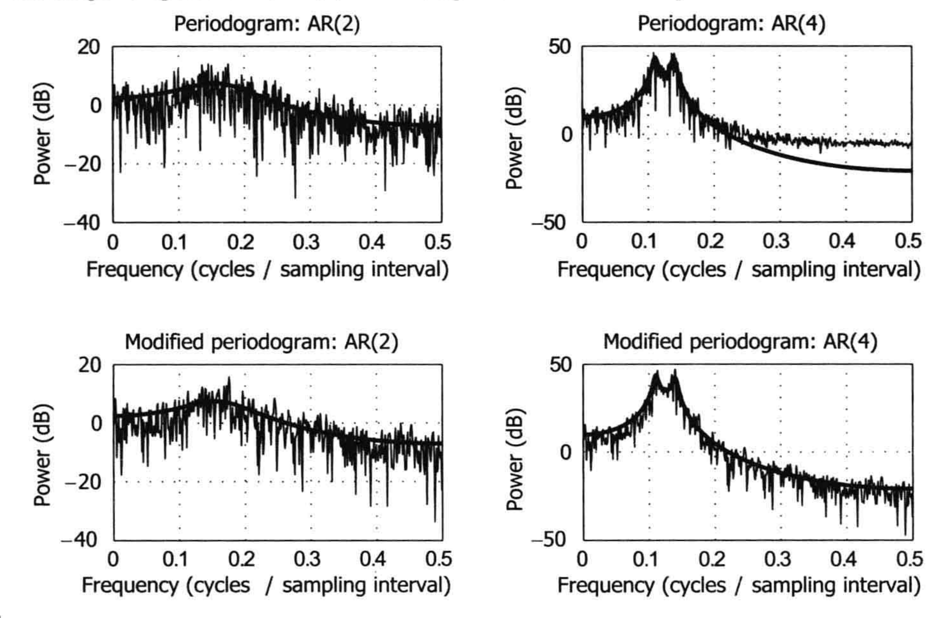


FIGURE 4.14
Illustration of properties of periodogram as a power spectrum estimator.

EXAMPLE 4.3.3 (FREQUENCY RESOLUTION PROPERTY OF THE PERIODOGRAM). Consider two unit-amplitude sinusoids observed in unit variance white noise. Let

$$x(n) = \cos(0.35\pi n + \varphi_1) + \cos(0.4\pi n + \varphi_2) + v(n)$$

where φ_1 and φ_2 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. Since two frequencies, 0.35π and 0.4π, are close, we will need (see Table 4.1)

$$N-1 > \frac{1.81\pi}{0.4\pi - 0.35\pi} \quad \text{or} \quad N > 37$$

To obtain a periodogram ensemble, 50 realizations of $x(n)$ for $N = 32$ and $N = 64$ were generated, and their periodograms were computed. The plots of these periodogram overlays and the corresponding ensemble average for $N = 32$ and $N = 64$ are shown in Figure 4.15. For $N = 32$, frequencies in the periodogram cannot be resolved, as expected; but for $N = 64$ it is possible to separate the two sinusoids with ease. Note that the modified periodogram (i.e., data windowing) will not help since windowing increases smoothing and smearing of peaks.

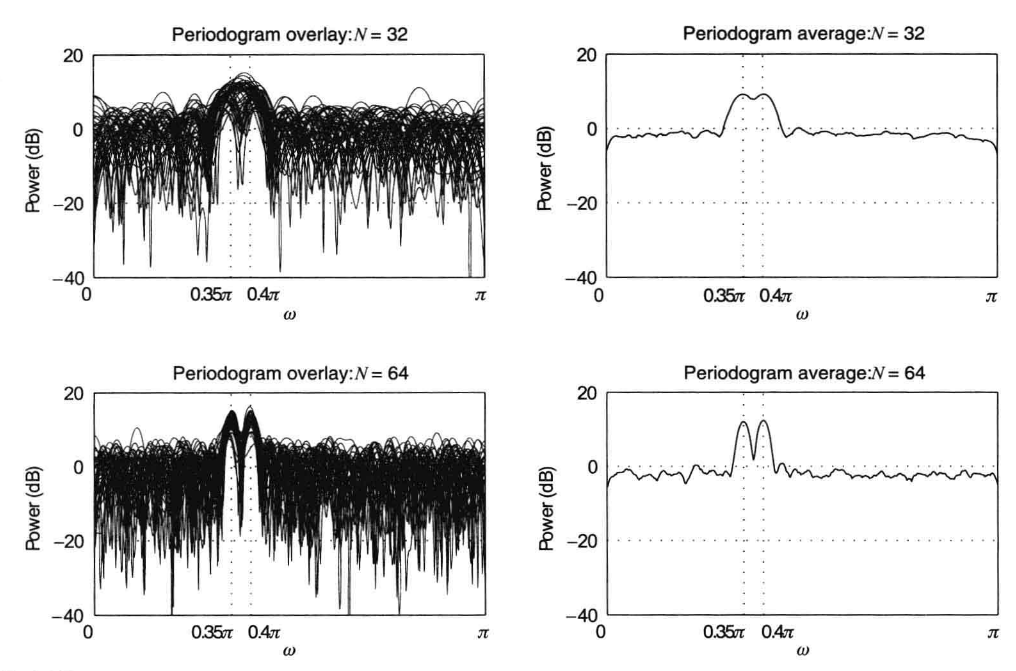


FIGURE 4.15
Illustration of the frequency resolution property of the periodogram in Example 4.3.3.

The case of nonzero mean. In the periodogram method of spectrum analysis in this section, we assumed that the random signal has zero mean. If a random signal has nonzero mean, it should be estimated using (2.4.20) and then removed from the signal prior to computing its periodogram. This is because the power spectrum of a nonzero mean signal has an impulse at the zero frequency. If this mean is relatively large, then because of the leakage inherent in the periodogram, this mean will obscure low-amplitude, low-frequency components of the spectrum. Even though the estimate is not an exact value, its removal often provides better estimates, especially at low frequencies.

Covariance of $\hat{R}_x(e^{j\omega})$. Obtaining an expression for the covariance of the periodogram is a rather complicated process. However, it has been shown (Jenkins and Watts 1968) that

$$\begin{aligned}&\operatorname{cov}\{\hat{R}_x(e^{j\omega_1}),\hat{R}_x(e^{j\omega_2})\}\\ &\quad\simeq R_x(e^{j\omega_1})R_x(e^{j\omega_2})\left(\left\{\frac{\sin[(\omega_1+\omega_2)N/2]}{N\sin[(\omega_1+\omega_2)/2]}\right\}^2+\left\{\frac{\sin[(\omega_1-\omega_2)N/2]}{N\sin[(\omega_1-\omega_2)/2]}\right\}^2\right)\end{aligned} \tag{4.3.26}$$

This expression applies to stationary random signals with zero mean and Gaussian probability density. The approximation becomes exact if the signal has a flat spectrum (white noise). Although this approximation deteriorates for non-Gaussian probability densities, the qualitative results that one can draw from this approximation appear to hold for a rather broad range of densities.

From (4.3.26), for $\omega_1=(2\pi/N)k_1$ and $\omega_2=(2\pi/N)k_2$ with k_1,k_2 integers, we have

$$\operatorname{cov}\{\hat{R}_x(e^{j\omega_1})\hat{R}_x(e^{j\omega_2})\}\simeq 0 \qquad \text{for } k_1\neq k_2 \tag{4.3.27}$$

Thus, values of the periodogram spaced in frequency by integer multiples of $2\pi/N$ are approximately uncorrelated. As the record length N increases, these uncorrelated periodogram samples come closer together, and hence the rate of fluctuations in the periodogram increases. This explains the results in Figure 4.12.

Variance of $\hat{R}_x(e^{j\omega})$. The variance of the periodogram at a particular frequency $\omega=\omega_1=\omega_2$ can be obtained from (4.3.26)

$$var\{\hat{R}_x(e^{j\omega})\}\simeq R_x^2(e^{j\omega})\left[1+\left(\frac{\sin\omega N}{N\sin\omega}\right)^2\right] \tag{4.3.28}$$

For large values of N, the variance of $\hat{R}_x(e^{j\omega})$ can be approximated by

$$var\{\hat{R}_x(e^{j\omega})\} \simeq \begin{cases} R_x^2(e^{j\omega}) & 0 < \omega < \pi \\ 2R_x^2(e^{j\omega}) & \omega = 0,\ \pi \end{cases} \tag{4.3.29}$$

This result is crucial, because it shows that the variance of the periodogram (estimate) remains at the level of $R_x^2(e^{j\omega})$ (quantity to be estimated), independent of the record length N used. Furthermore, since the variance does not tend to zero as $N \to \infty$, the periodogram is not a consistent estimator; that is, its distribution does not tend to cluster more closely around the true spectrum as N increases.[4]

This behavior was illustrated in Example 4.3.1.The variance of $\hat{R}_x(e^{j\omega_k})$ fails to decrease as N increases because the number of periodogram values $\hat{R}_x(e^{j\omega_k})$, $k = 0, 1, \cdots, N-1$, is always equal to the length N of the data record.

EXAMPLE 4.3.4 (COMPARISON OF PERIODOGRAM AND MODIFIED PERIODOGRAM)

Consider the case of three sinusoids discussed in Section 4.1.4. In particular, we assume that these sinusoids are observed in white noise with

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_1) + 0.25\cos(0.8\pi n + \phi_1) + v(n)$$

where ϕ_1, ϕ_2, and ϕ_3 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using N=128. The periodograms and the Hamming window-based modified periodograms of these realizations were computed, and the results are shown in Figure 4.16. The top row of the figure contains periodogram overlays and the corresponding ensemble average for the unwindowed periodogram, and the bottom row shows the same for the modified periodogram. Spurious peaks (especially near the two close frequencies) in the periodogram have been suppressed by the data windowing poeration in the modified periodogram; hence the peak corresponding to 0.8π is sufficiently enhanced. This enhancement is cleary at the expense of the frequency resolution (or smearing of the true peaks), which is to be expected. The overall variance of the noise floor is still not reduced.

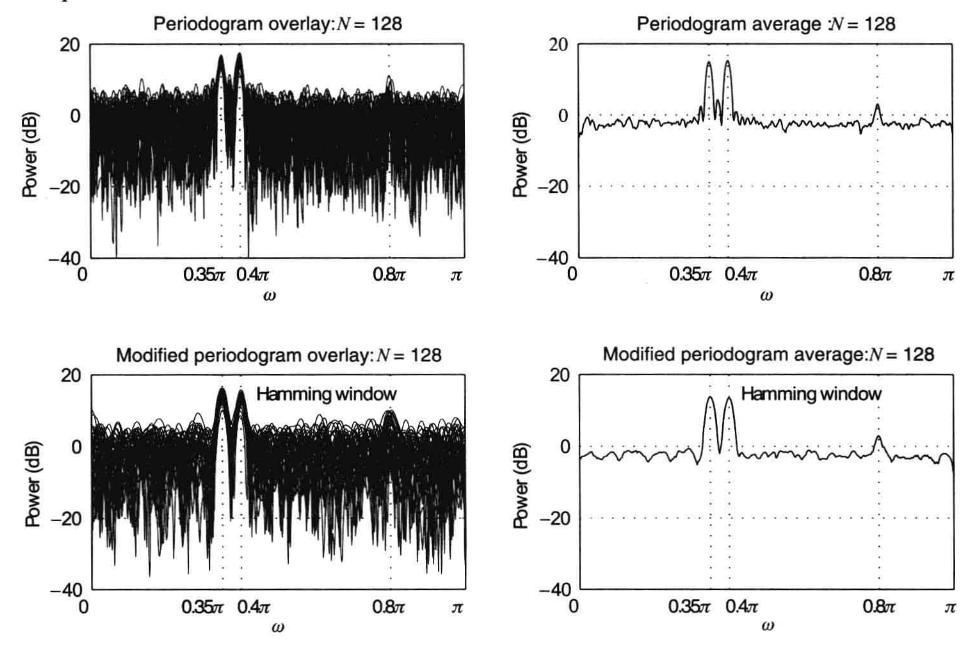


FIGURE 4.16
Comparison of periodogram and modified periodogram in Example 4.3.4.

Failure of the periodogram

To conclude, we note that the periodogram in its "basic form" is a very poor estimator of the power spectrum

[4] The definition of the PSD by $R_x(e^{j\omega}) = \lim_{N\to\infty} \hat{R}_x(e^{j\omega})$ is not valid because even if $\lim_{N\to\infty} E\{\hat{R}_x(e^{j\omega})\} = R_x(e^{j\omega})$, the variance of $\hat{R}_x(e^{j\omega})$ does not tend to zero as $N \to \infty$ (Papoulis 1991).

function. The failure of the periodogram when applied to random signals is uniquely pointed out in Jenkins and Watts (1968, p. 213):

> The basic reason why Fourier analysis breaks down when applied to time series is that it is based on the assumption of *fixed* amplitudes, frequencies and phases. Time series, on the other hand, are characterized by *random* changes of frequencies, amplitudes and phases. Therefore it is not surprising that Fourier methods need to be adapted to account for the random nature of a time series.

The attempt at improving the periodogram by windowing the available data, that is, by using the modified periodogram in Example 4.3.4, showed that the presence and the length of the window had no effect on the variance. The major problems with the periodogram lie in its variance, which is on the order of $R_x^2(\mathrm{e}^{\mathrm{j}\omega})$, as well as in its erratic behavior. Thus, to obtain a better estimator, we should reduce its variance; that is, we should "smooth" the periodogram. From the previous discussion, it follows that the sequence $\tilde{\hat{R}}_x(k)$, k=0, 1, ⋯, $N-1$, of the harmonic periodogram components can be reasonably assumed to be a sequence of uncorrelated random variables. Furthermore, it is well known that the variance of the sum of K uncorrelated random variables with the same variance is $1/K$ times the variance of one of these individual random variables. This suggests two ways of reducing the variance, which also lead to smoother spectral estimators:

- Average contiguous values of the periodogram.
- Average periodograms obtained from multiple data segments.

It should be apparent that owing to stationarity, the two approaches should provide comparable results under similar circumstances.

4.3.2 Power Spectrum Estimation by Smoothing a Single Periodogram—The Blackman-Tukey Method

The idea of reducing the variance of the periodogram through smoothing using a moving-average filter was first proposed by Daniel (1946). The estimator proposed by Daniel is a zero-phase moving-average filter, given by

$$\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega_k}) \triangleq \frac{1}{2M+1}\sum_{j=-M}^{M}\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega_{k-j}}) \triangleq \sum_{j=-M}^{M} W(\mathrm{e}^{\mathrm{j}\omega_j})\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega_{k-j}}) \tag{4.3.30}$$

where $\omega_k = (2\pi/N)k,\ k = 0,\ 1,\ \cdots,\ N-1$, $W(e^{j\omega_j}) \triangleq 1/(2M+1)$, and the superscript (PS) denotes periodogram smoothing. Since the samples of the periodogram are approximately uncorrelated,

$$var\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega_k})\} \simeq \frac{1}{2M+1} var\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega_k})\} \tag{4.3.31}$$

that is, averaging $2M+1$ consecutive spectral lines reduces the variance by a factor of $2M+1$. The quantity $\Delta\omega \approx (2\pi/N)(2M+1)$ determines the frequency resolution, since any peaks within the $\Delta\omega$ range are smoothed over the entire interval $\Delta\omega$ into a single peak and cannot be resolved. Thus, increasing M reduces the variance (resulting in a smoother spectrum estimate), at the expense of spectral resolution. This is the fundamental tradeoff in practical spectral analysis.

Blackman-Tukey approach

The discrete moving average in (4.3.30) is computed in the frequency domain. We now introduce a better and simpler way to smooth the periodogram by operating on the estimated autocorrelation sequence. To this end, we note that the continuous frequency equivalent of the discrete convolution formula (4.3.30) is the periodic convolution

$$\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\hat{R}_x(\mathrm{e}^{\mathrm{j}(\omega-\theta)})W_a(\mathrm{e}^{\mathrm{j}\theta})\,\mathrm{d}\theta = \hat{R}_x(\mathrm{e}^{\mathrm{j}\omega}) \otimes W_a(\mathrm{e}^{\mathrm{j}\omega}) \tag{4.3.32}$$

where $W_a(\mathrm{e}^{\mathrm{j}\omega})$ is a periodic function of ω with period 2π, given by

$$W_a(\mathrm{e}^{\mathrm{j}\omega}) = \begin{cases} \dfrac{1}{\Delta\omega} & |\omega| < \dfrac{\Delta\omega}{2} \\ 0 & \dfrac{\Delta\omega}{2} \le \omega \le \pi \end{cases} \tag{4.3.33}$$

By using the convolution theorem, (4.3.32) can be written as

$$\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{l=-(L-1)}^{L-1} \hat{r}_x(l) w_a(l) \mathrm{e}^{-\mathrm{j}\omega l} \tag{4.3.34}$$

where $w_a(l)$ is the inverse Fourier transform of $W_a(\mathrm{e}^{\mathrm{j}\omega})$ and $L < N$. As we have already mentioned, the window $w_a(l)$ is known as the correlation or lag window.[5] The correlation window corresponding to (4.3.33) is

$$w_a(l) = \frac{\sin(l\Delta\omega/2)}{\pi l} \qquad -\infty < l < \infty \tag{4.3.35}$$

Since $w_a(l)$ has infinite duration, its truncation at $|l| = L \le N$ creates ripples in $W_a(\mathrm{e}^{\mathrm{j}\omega})$ (Gibbs effect). To avoid this problem, we use correlation windows with finite duration, that is, $w_a(l) = 0$ for $|l| > L \le N$. For real sequences, where $\hat{r}_x(l)$ is real and even, $w_a(l)$ [and hence $W_a(\mathrm{e}^{\mathrm{j}\omega})$] should be real and even. Given that $\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})$ is nonnegative, a sufficient (but not necessary) condition that $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$ be nonnegative is that $W_a(\mathrm{e}^{\mathrm{j}\omega}) \ge 0$ for all ω. This condition holds for the Bartlett (triangular) and Parzen (see Problem 4.11) windows, but it does not hold for the Hamming, Hanning, or Kaiser window.

Thus, we note that smoothing the periodogram $\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})$ by convolving it with the spectrum $W_a(\mathrm{e}^{\mathrm{j}\omega}) = F\{w_a(l)\}$ is equivalent to windowing the autocorrelation estimate $\hat{r}_x(l)$ with the correlation window $w_a(l)$. This approach to power spectrum estimation, which was introduced by Blackman and Tukey (1959), involves the following steps:

1. Estimate the autocorrelation sequence from the unwindowed data.
2. Window the obtained autocorrelation samples.
3. Compute the DTFT of the windowed autocorrelation as given in (4.3.34).

A pictorial comparison between the theoretical [i.e., using (4.3.32)] and the above practical computation of power spectrum using the single-periodogram smoothing is shown in Figure 4.17.

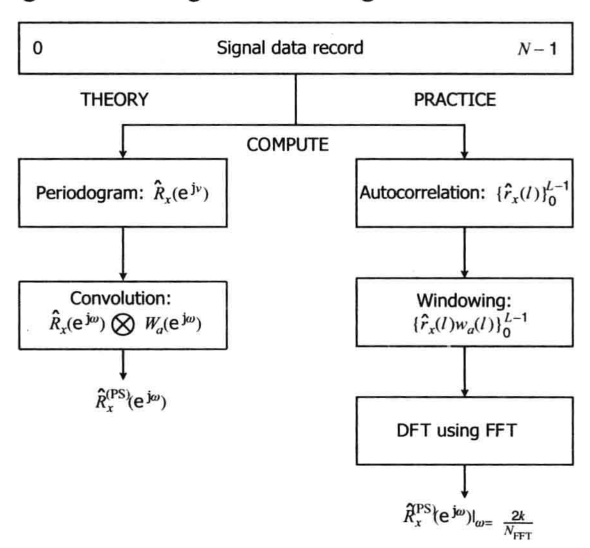


FIGURE 4.17
Comparison of the theory and practice of the Blackman-Tukey method.

The resolution of the Blackman-Tukey power spectrum estimator is determined by the duration $2L-1$ of the correlation window. For most correlation windows, the resolution is measured by the 3-dB bandwidth of the mainlobe, which is on the order of $2\pi/L$ rad per sampling interval.

The statistical quality of the Blackman-Tukey estimate $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$ can be evaluated by examining its mean, covariance, and variance.

Mean of $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$. The expected value of the smoothed periodogram $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$ can be obtained by using (4.3.34) and (4.2.11). Indeed, we have

[5] The term *spectral window* is quite often used for $W_a(\mathrm{e}^{\mathrm{j}\omega}) = F\{w_a(l)\}$, the Fourier transform of the correlation window. However, this term is misleading because $W_a(\mathrm{e}^{\mathrm{j}\omega})$ is essentially a frequency-domain impulse response. We use the term *correlation window* for $w_a(l)$ and the term *Fourier transform of the correlation window* for $W_a(\mathrm{e}^{\mathrm{j}\omega})$.

$$\begin{aligned} E\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} &= \sum_{l=-(L-1)}^{L-1} E\{\hat{r}_x(l)\}w_a(l)\mathrm{e}^{-\mathrm{j}\omega l} \\ &= \sum_{l=-(L-1)}^{L-1} r_x(l)\left(1-\frac{|l|}{N}\right)w_a(l)\mathrm{e}^{-\mathrm{j}\omega l} \end{aligned} \tag{4.3.36}$$

or, using the frequency convolution theorem, we have

$$E\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} = R_x(\mathrm{e}^{\mathrm{j}\omega})\otimes W_B(\mathrm{e}^{\mathrm{j}\omega})\otimes W_a(\mathrm{e}^{\mathrm{j}\omega}) \tag{4.3.37}$$

where
$$W_B(\mathrm{e}^{\mathrm{j}\omega}) = F\left\{\left(1-\frac{|l|}{N}\right)w_R(n)\right\} = \frac{1}{N}\left[\frac{\sin(\omega N/2)}{\sin(\omega/2)}\right]^2 \tag{4.3.38}$$

is the Fourier thansform of the Bartlett window. Since $E\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} \neq R_x(\mathrm{e}^{\mathrm{j}\omega})$, $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$ is a *biased* estimate of $R_x(\mathrm{e}^{\mathrm{j}\omega})$.

For $L \ll N, (1-|l|/N) \simeq 1$ and hence we obtain

$$\begin{aligned} E\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} &= \sum_{l=-(L-1)}^{L-1} r_x(l)\left(1-\frac{|l|}{N}\right)w_a(l)\mathrm{e}^{-\mathrm{j}\omega l} \\ &\simeq R_x(\mathrm{e}^{\mathrm{j}\omega})\otimes W_a(\mathrm{e}^{\mathrm{j}\omega}) \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} R_x(\mathrm{e}^{\mathrm{j}\theta})W_a(\mathrm{e}^{\mathrm{j}(\omega-\theta)})\,\mathrm{d}\theta \end{aligned} \tag{4.3.39}$$

If L is sufficiently large, the correlation window $w_a(l)$ consists of a narrow mainlobe. If $R_x(\mathrm{e}^{\mathrm{j}\omega})$ can be assumed to be constant within the mainlobe, we have

$$E\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} \simeq R_x(\mathrm{e}^{\mathrm{j}\omega})\frac{1}{2\pi}\int_{-\pi}^{\pi} W_a(\mathrm{e}^{\mathrm{j}(\omega-\theta)})\,\mathrm{d}\theta$$

which implies that $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$ is *asymptotically unbiased* if

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} W_a(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega = w_a(0) = 1 \tag{4.3.40}$$

that is, if the spectrum of the correlation window has unit area. Under this condition, if both L and N tend to infinity, then $W_a(\mathrm{e}^{\mathrm{j}\omega})$ and $W_B(\mathrm{e}^{\mathrm{j}\omega})$ become periodic impulse trains and the convolution (4.3.37) reproduces $R_x(\mathrm{e}^{\mathrm{j}\omega})$.

Covariance of $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$. The following approximation

$$\operatorname{cov}\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega_1}), \hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega_2})\} \simeq \frac{1}{2\pi N}\int_{-\pi}^{\pi} R_x^2(\mathrm{e}^{\mathrm{j}\theta})W_a(\mathrm{e}^{\mathrm{j}(\omega_1-\theta)})W_a(\mathrm{e}^{\mathrm{j}(\omega_2-\theta)})\,\mathrm{d}\theta \tag{4.3.41}$$

derived in Jenkins and Watts (1968), holds under the assumptions that (1) N is sufficiently large that $W_B(\mathrm{e}^{\mathrm{j}\omega})$ behaves as a periodic impulse train and (2) L is sufficiently large that $W_a(\mathrm{e}^{\mathrm{j}\omega})$ is sufficiently narrow that the product $W_a(\mathrm{e}^{\mathrm{j}(\omega_1+\theta)})W_a(\mathrm{e}^{\mathrm{j}(\omega_2-\theta)})$ is negligible. Hence, the covariance increases proportionally to the width of $W_a(\mathrm{e}^{\mathrm{j}\omega})$, and the amount of overlap between the windows $W_a(e^{j(\omega_1-\theta)})$ (centered at ω_1) and $W_a(\mathrm{e}^{\mathrm{j}(\omega_2-\theta)})$ (centered at ω_2) increases.

Variance of $\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})$. When $\omega = \omega_1 = \omega_2$, (4.3.41) gives

$$\operatorname{var}\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} \simeq \frac{1}{2\pi N}\int_{-\pi}^{\pi} R_x^2(\mathrm{e}^{\mathrm{j}\omega})W_a^2(\mathrm{e}^{\mathrm{j}(\omega-\theta)})\,\mathrm{d}\theta \tag{4.3.42}$$

If $R_x(\mathrm{e}^{\mathrm{j}\omega})$ is smooth within the width of $W_a(\mathrm{e}^{\mathrm{j}\omega})$, then

$$\operatorname{var}\{\hat{R}_x^{(\mathrm{PS})}(\mathrm{e}^{\mathrm{j}\omega})\} \simeq R_x^2(\mathrm{e}^{\mathrm{j}\omega})\frac{1}{2\pi N}\int_{-\pi}^{\pi} W_a^2(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega \tag{4.3.43}$$

or

$$\text{var}\{\hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega})\} \simeq \frac{E_w}{N} R_x^2(\text{e}^{\text{j}\omega}) \qquad 0<\omega<\pi \tag{4.3.44}$$

where
$$E_w = \frac{1}{2\pi}\int_{-\pi}^{\pi} W_a^2(\text{e}^{\text{j}\omega})\text{d}\omega = \sum_{l=-(L-1)}^{L-1} w_a^2(l) \tag{4.3.45}$$

is the energy of the correlation window. From (4.3.29) and (4.3.44) we have

$$\frac{\text{var}\{\hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega})\}}{\text{var}\{\hat{R}_x(\text{e}^{\text{j}\omega})\}} \simeq E_w / N \qquad 0<\omega<\pi \tag{4.3.46}$$

which is known as the *variance reduction factor* or *variance ratio* and provides the reduction in variance attained by smoothing the periodogram.

In the beginning of this section, we explained the variance reduction in terms of frequency-domain averaging. An alternative explanation can be provided by considering the windowing of the estimated autocorrelation. As discussed in Section 4.2 the variance of the autocorrelation estimate increases as $|l|$ approaches N because fewer and fewer samples are used to compute the estimate. Since every value of $\hat{r}_x(l)$ affects the value of $\hat{R}_x(\omega)$ at all frequencies, the less reliable values affect the quality of the periodogram everywhere. Thus, we can reduce the variance of the periodogram by minimizing the contribution of autocorrelation terms with large variance, that is, with lags close to N, by proper windowing.

As we have already stressed, there is a tradeoff between resolution and variance. For the variance to be small, we must choose a window that contains a small amount of energy E_w. Since $|w_a(l)| \le 1$, we have $E_w \le 2L$. Thus, to reduce the variance, we must have $L \ll N$. The bias of $\hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega})$ is directly related to the resolution, which is determined by the mainlobe width of the window, which in turn is proportional to $1/L$. Hence, to reduce the bias, $W_a(\text{e}^{\text{j}\omega})$ should have a narrow mainlobe that demands a large L. The requirements for high resolution (small bias) and low variance can be simultaneously satisfied only if N is sufficiently large. The variance reduction for some commonly used windows is examined in Problem 4.12. Empirical evidence suggests that use of the Parzen window is a reasonable choice.

Confidence intervals. In the interpretation of spectral estimates, it is important to know whether the spectral details are real or are due to statistical fluctuations. Such information is provided by the confidence intervals (Chapter 2). When the spectrum is plotted on a logarithmic scale, the $(1-\alpha)\times 100$ percent confidence interval is constant at every frequency, and it is given by (Koopmans 1974)

$$\left(10\log \hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega}) - 10\log\frac{\chi_\nu^2(1-\alpha/2)}{\nu}, 10\log \hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega}) + 10\log\frac{\nu}{\chi_\nu^2(\alpha/2)}\right) \tag{4.3.47}$$

where
$$\nu = \frac{2N}{\sum_{k=-(L-1)}^{L} w_a^2(l)} \tag{4.3.48}$$

is the degress of freedom of a χ_ν^2 distribution.

Computation of $\hat{R}_x^{(\text{PS})}(\text{e}^{\text{j}\omega})$ ***using the DFT***. In practice, the Blackman-Tukey power spectrum estimator is computed by using an *N*-point DFT as follows:

1. Estimate the autocorrelation $r_x(l)$, using the formula

$$\hat{r}_x(l) = \hat{r}_x^*(-l) = \frac{1}{N}\sum_{n=0}^{N+l-1} x(n+l)x^*(n) \qquad l=0,\ 1,\ \cdots,\ L-1 \tag{4.3.49}$$

For $L>100$, indirect computation of $\hat{r}_x(l)$ by using DFT techniques is usually more efficient (see Problem 4.13).

2. Form the sequence

$$f(l) = \begin{cases} \hat{r}_x(l)w_a(l) & 0 \le l \le L-1 \\ 0 & L \le l \le N-L \\ \hat{r}_x^*(N-l)w_a(N-l) & N-L+1 \le l \le N-1 \end{cases} \tag{4.3.50}$$

3. Compute the power spectrum estimate

$$\hat{R}_x^{(\mathrm{PS})}(e^{j\omega})\big|_{\omega=(2\pi/N)k} = F(k) = \mathrm{DFT}\{f(l)\} \quad 0 \le k \le N-1 \tag{4.3.51}$$

as the N-point DFT of the sequence $f(l)$.

MATLAB does not provide a direct function to implement the Blackman-Tukey method. However, such a function can be easily constructed by using built-in MATLAB functions and the above approach. The book toolbox function

```
Rx = bt_psd(x, Nfft, window, L);
```

implements the above algorithm in which `window` is any available MATLAB window and `Nfft` is chosen to be larger than N to obtain a high-density spectrum.

EXAMPLE 4.3.5 (BLACKMAN-TUKEY METHOD)]. Consider the spectrum estimation of three sinusoids in white noise given in Example 4.3.4, that is,

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25\cos(0.8\pi n + \phi_3) + v(n) \tag{4.3.52}$$

where ϕ_1, ϕ_2, and ϕ_3 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using $N = 512$. The autocorrelations of these realizations were estimated up to lag $L = 64$, 128, and 256. These autocorrelations were windowed using the Bartlett window, and then their 1024-point DFT was computed as the spectrum estimate. The results are shown in Figure 4.18. The top row of the figure contains estimate overlays and the corresponding ensemble average for $L = 64$, the middle row for $L = 128$, and the bottom row for $L = 256$. Several observations can be made from these plots. First, the variance in the estimate has considerably reduced over the periodogram estimate. Second, the lower the lag distance L, the lower the variance and the resolution (i.e., the higher the smoothing of the peaks). This observation is consistent with our discussion above about the effect of L on the quality of estimates. Finally, all the frequencies including the one at 0.8π are clearly distinguishable, something that the basic periodogram could not achieve.

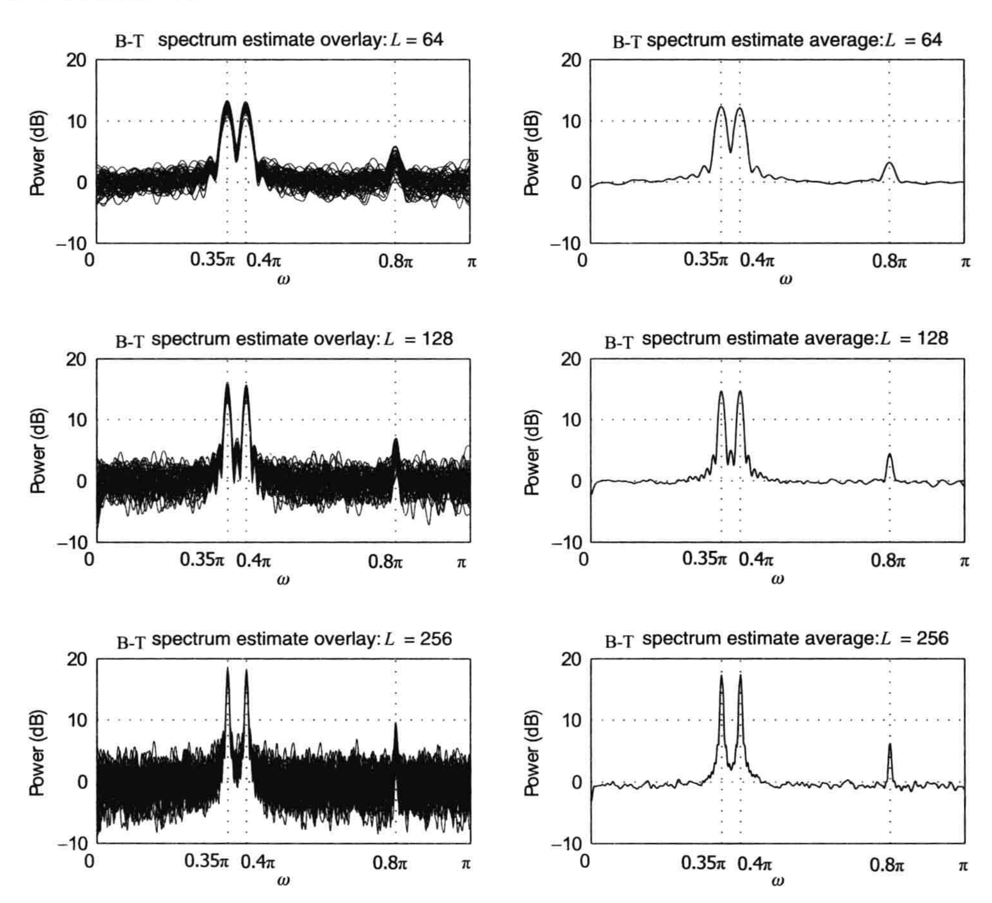


FIGURE 4.18
Spectrum estimation of three sinusoids in white noise using the Blackman-Tukey method in Example 4.3.5.

4.3.3 Power Spectrum Estimation by Averaging Multiple Periodograms—The Welch-Bartlett Method

As mentioned in Section 4.3.1, in general, the variance of the sum of K IID random variables is $1/K$ times the variance of each of the random variables. Thus, to reduce the variance of the periodogram, we could average the periodograms from K different realizations of a stationary random signal. However, in most practical applications, only a single realization is available. In this case, we can subdivide the existing record $\{x(n), 0 \le n \le N-1\}$ into K (possibly overlapping) smaller segments as follows:

$$x_i(n) = x(iD+n)w(n) \qquad 0 \le n \le L-1, \ 0 \le i \le K-1 \tag{4.3.53}$$

where $w(n)$ is a window of duration L and D is an *offset* distance. If $D < L$, the segments overlap; and for $D = L$, the segments are contiguous. The periodogram of the i th segment is

$$\hat{R}_{x,i}(\mathrm{e}^{\mathrm{j}\omega}) \triangleq \frac{1}{L} |X_i(\mathrm{e}^{\mathrm{j}\omega})|^2 = \frac{1}{L}\left|\sum_{n=0}^{L-1} x_i(n)\mathrm{e}^{-\mathrm{j}\omega n}\right|^2 \tag{4.3.54}$$

We remind the reader that the window $w(n)$ in (4.3.53) is called a data window because it is applied directly to the data, in contrast to a correlation window that is applied to the autocorrelation sequence [see (4.3.34)]. Notice that there is no need for the data window to have an even shape or for its Fourier transform to be nonnegative. The purpose of using the data window is to control spectral leakage.

The spectrum estimate $\hat{R}_x^{(PA)}(e^{j\omega})$ is obtained by averaging K periodograms as follows:

$$\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega}) \triangleq \frac{1}{K}\sum_{i=0}^{K-1} \hat{R}_{x,i}(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{K}\sum_{i=0}^{K-1} |X_i(\mathrm{e}^{\mathrm{j}\omega})|^2 \tag{4.3.55}$$

where the superscript (PA) denotes periodogram averaging. To determine the bias and variance of $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$, we let $D = L$ so that the segments do not overlap. The so-computed estimate $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$ is known as the *Bartlett* estimate. We also assume that $r_x(l)$ is very small for $|l| > L$. This implies that the signal segments can be assumed to be approximately uncorrelated. To show that the simple periodogram averaging in Bartlett's method reduces the periodogram variance, we consider the following example.

EXAMPLE 4.3.6 (PERIODOGRAM AVERAGING). Let $x(n)$ be a stationary white Gaussian noise with zero mean and unit variance. The theoretical spectrum of $x(n)$ is

$$R_x(\mathrm{e}^{\mathrm{j}\omega}) = \sigma_x^2 = 1 \qquad -\pi < \omega \le \pi$$

An ensemble of 50 different 512-point records of $x(n)$ was generated using a pseudorandom number generator. The Bartlett estimate of each record was computed for $K = 1$ (i.e., the basic periodogram), $K = 4$ (or $L = 128$), and $K = 8$ (or $L = 64$). The results in the form of estimate overlays and averages are shown in Figure 4.19. The effect of periodogram averaging is clearly evident.

Mean of $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$. The mean value of $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$ is

$$E\{\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})\} = \frac{1}{K}\sum_{i=0}^{K-1} E\{\hat{R}_{x,i}(\mathrm{e}^{\mathrm{j}\omega})\} = E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\} \tag{4.3.56}$$

where we have assumed that $E\{\hat{R}_{x,i}(\mathrm{e}^{\mathrm{j}\omega})\} = E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\}$ because of the stationarity assumption. From (4.3.56) and (4.3.15), we have

$$E\{\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})\} = E\{\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})\} = \frac{1}{2\pi L}\int_{-\pi}^{\pi} R_x(\mathrm{e}^{\mathrm{j}\theta}) R_w(\mathrm{e}^{\mathrm{j}(\omega-\theta)})\,\mathrm{d}\theta \tag{4.3.57}$$

where $R_w(\mathrm{e}^{\mathrm{j}\omega})$ is the spectrum of the data window $w(n)$. Hence, $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$ is a biased estimate of $R_x(\mathrm{e}^{\mathrm{j}\omega})$. However, if the data window is normalized such that

$$\sum_{n=0}^{L-1} w^2(n) = L \tag{4.3.58}$$

the estimate $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$ becomes asymptotically unbiased [see the discussion following equation (4.3.15)].

Variance of $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$. The variance of $\hat{R}_x^{(\mathrm{PA})}(\mathrm{e}^{\mathrm{j}\omega})$ is

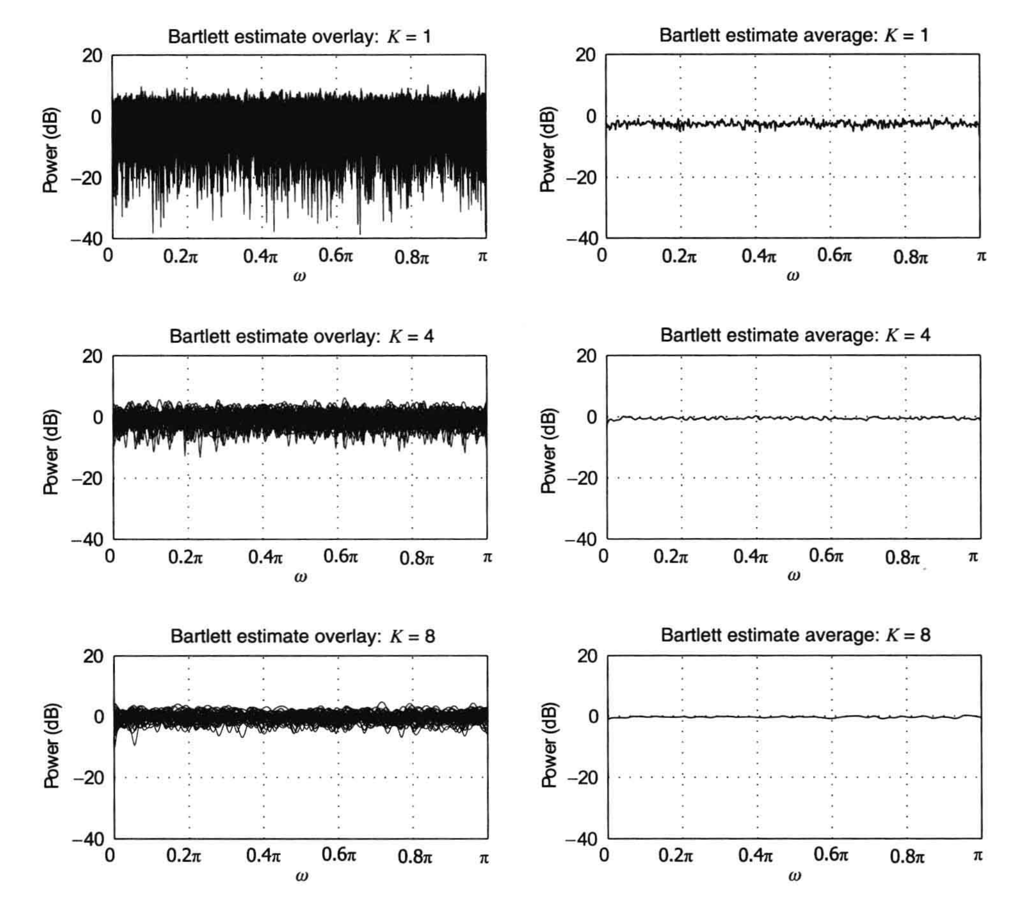


FIGURE 4.19
Spectral estimation of white noise using Bartlett's method in Example 4.3.6.

$$\operatorname{var}\{\hat{R}_x^{(\mathrm{PA})}(e^{j\omega})\} = \frac{1}{K}\operatorname{var}\{\hat{R}_x(e^{j\omega})\} \tag{4.3.59}$$

or using (4.3.29) gives

$$\operatorname{var}\{\hat{R}_x^{(\mathrm{PA})}(e^{j\omega})\} \simeq \frac{1}{K}R_x^2(e^{j\omega}) \tag{4.3.60}$$

Clearly, as K increases, the variance tends to zero. Thus, $\hat{R}_x^{(\mathrm{PA})}(e^{j\omega})$ provides an asymptotically unbiased and consistent estimate of $R_x(e^{j\omega})$. If N is fixed and $N = KL$, we see that increasing K to reduce the variance (or equivalently obtain a smoother estimate) results in a decrease in L, that is, a reduction in resolution (or equivalently an increase in bias).

When $w(n)$ in (4.3.53) is the rectangular window of duration L, the square of its Fourier transform is equal to the Fourier transform of the triangular sequence $w_T(n) \triangleq L - |l|, |l| < L$, which when combined with the $1/L$ factor in (4.3.57), results in the Bartlett window

$$w_B(l) = \begin{cases} 1 - |l|/L & |l| < L \\ 0 & \text{elsewhere} \end{cases} \tag{4.3.61}$$

with

$$W_B(e^{j\omega}) = \frac{1}{L}\left[\frac{\sin(\omega L/2)}{\sin(\omega/2)}\right]^2 \tag{4.3.62}$$

This special case of averaging multiple nonoverlapping periodograms was introduced by Bartlett (1953).

The method has been extended to modified overlapping periodograms by Welch (1970), who has shown that the

shape of the window does not affect the variance formula (4.3.59). Welch showed that overlapping the segments by 50 percent reduces the variance by about a factor of 2, owing to doubling the number of segments. More overlap does not result in additional reduction of variance because the data segments become less and less independent. Clearly, the nonoverlapping segments can be uncorrelated only for white noise signals. However, the data segments can be considered approximately uncorrelated if they do not have sharp spectral peaks or if their autocorrelations decay fast.

Thus, the variance reduction factor for the spectral estimator $\hat{R}_x^{(PA)}(e^{j\omega})$ is

$$\frac{\text{var}\{\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})\}}{\text{var}\{\hat{R}_x(\text{e}^{\text{j}\omega})\}} \simeq 1/K \qquad 0<\omega<\pi \tag{4.3.63}$$

and is reduced by a factor of 2 for 50 percent overlap.

Confidence intervals. The $(1-\alpha)\times 100$ percent confidence interval on a logarithmic scale may be shown to be (Jenkins and Watts 1968)

$$\left(10\log\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})-10\log\frac{\chi_{2K}^2(1-\alpha/2)}{2K},\ 10\log\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})+10\log\frac{2K}{\chi_{2K}^2(\alpha/2)}\right) \tag{4.3.64}$$

where χ_{2K}^2 is a chi-squared distribution with $2K$ degrees of freedom.

Computation of $\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})$ using the DFT. In practice, to compute $\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})$ at L equally spaced frequencies $\omega_k = 2\pi k/L, 0\le k\le L-1$, the method of periodogram averaging can be easily and efficiently implemented by using the DFT as follows (we have assumed that L is even):

1. Segment data $\{x(n)\}_0^{N-1}$ into K segments of length L, each offset by D duration using

$$\bar{x}_i(n) = x(iD+n) \qquad 0\le i\le K-1,\ 0\le n\le L-1 \tag{4.3.65}$$

If $D=L$, there is no overlap; and if $D=L/2$, the overlap is 50 percent.

2. Window each segment, using data window $w(n)$

$$x_i(n)=\bar{x}_i(n)w(n)=x(iD+n)w(n) \qquad 0\le i\le K-1,\ 0\le n\le L-1 \tag{4.3.66}$$

3. Compute the N-point DFTs $X_i(k)$ of the segments $x_i(n)$, $0\le i\le K-1$,

$$\tilde{X}_i(k)=\sum_{n=0}^{L-1}x_i(n)\text{e}^{-\text{j}(2\pi/L)kn} \qquad 0\le k\le L-1,\ 0\le i\le K-1 \tag{4.3.67}$$

4. Accumulate the squares $|\tilde{X}_i(k)|^2$

$$\tilde{S}_i(k)\triangleq\sum_{i=0}^{K-1}|\tilde{X}_i(k)|^2 \qquad 0\le k\le L/2 \tag{4.3.68}$$

5. Finally, normalize by KL to obtain the estimate $\hat{R}_x^{(\text{PA})}(k)$:

$$\hat{R}_x^{(\text{PA})}(k)=\frac{1}{KL}\sum_{i=0}^{K-1}\tilde{S}_i(k) \qquad 0\le k\le N/2 \tag{4.3.69}$$

At this point we emphasize that the spectrum estimate $\hat{R}_x^{(\text{PA})}(k)$ is always nonnegative. A pictorial description of this computational algorithm is shown in Figure 4.20. A more efficient way to compute $\hat{R}_x^{(\text{PA})}(k)$ is examined in Problem 4.14.

In MATLAB the Welch-Bartlett method is implemented by using the function

Rx = psd(x, Nfft, Fs, *windows*(L), Noverlap, 'none');

where `window` is the name of any MATLAB-provided window function (e.g., `hamming`); `Nfft` is the size of the DFT, which is chosen to be larger than L to obtain a high-density spectrum; `Fs` is the sampling frequency, which is used for plotting purposes; and `Noverlap` specifies the number of overlapping samples. If the `boxcar` window is used along with `Noverlap=0`, then we obtain Bartlett's method of periodogram averaging. (Note that `Noverlap` is different from the offset parameter D given above.) If `Noverlap=L/2` is used, then we obtain Welch's averaged periodogram method with 50 percent overlap.

A biased estimate $\hat{r}_x(l)$, $|l|<L$, of the autocorrelation sequence of $x(n)$ can be obtained by taking the inverse N-point DFT of $\hat{R}_x^{(\text{PA})}(k)$ if $N\ge 2L-1$. Since only samples of the continuous spectrum $\hat{R}_x^{(\text{PA})}(\text{e}^{\text{j}\omega})$ are available, the obtained autocorrelation sequence $\hat{r}_x^{(\text{PA})}(l)$ is an aliased version of the true autocorrelation $r_x(l)$ of

the signal $x(n)$ (see Problem 4.14).

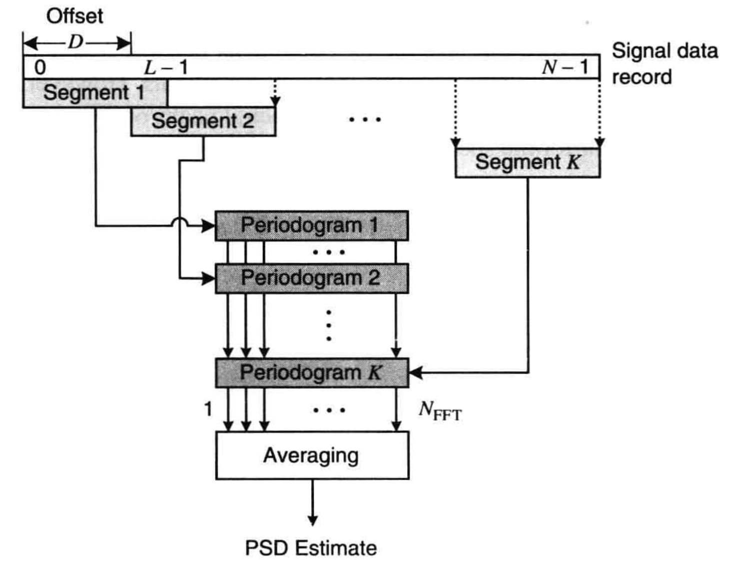


FIGURE 4.20
Pictorial description of the Welch-Bartlett method.

EXAMPLE 4.3.7 (BARTLETT'S METHOD) Consider again the spectrum estimation of three sinusoids in white noise given in Example 4.3.4, that is,

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25\cos(0.8\pi n + \phi_3) + v(n) \tag{4.3.70}$$

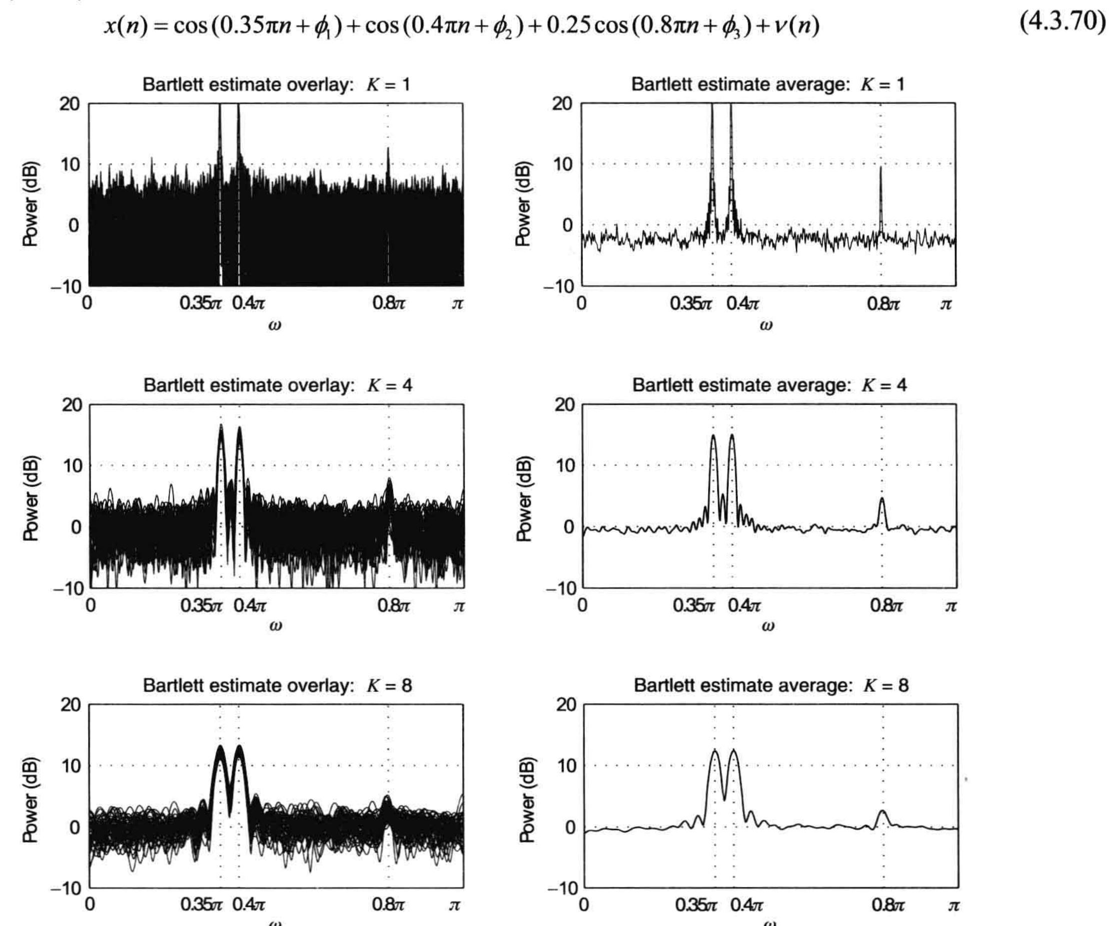


FIGURE 4.21
Estimation of three sinusoids in white noise using Bartlett's method in Example 4.3.7.

where φ_1, φ_2, and φ_3 are jointly independent random variables uniformly distributed over $[-\pi,\pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using $N=512$. The Bartlett estimate of each ensemble was computed for $K=1$ (i.e., the basic periodogram), $K=4$ (or $L=128$), and $K=8$ (or $L=64$). The results in the form of estimate overlays and averages are shown in Figure 4.21. Observe that the variance in the estimate has consistently reduced over the periodogram estimate as the number of averaging segments has increased. However, this reduction has come at the price of broadening of the spectral peaks. Since no window is used, the sidelobes are very prominent even for the $L=8$ segment. Thus confidence in the $\omega=0.8\pi$ spectral line is not very high for the $L=8$ case.

EXAMPLE 4.3.8 (WELCH'S METHOD). Consider Welch's method for the random process in the above example for $N=512$, 50 percent overlap, and a Hamming window. Three different values for L were considered; $L=256$ (3 segments), $L=128$ (7 segments), and $L=64$ (15 segments). The estimate overlays and averages are shown in Figure 4.22. In comparing these results with those in Figure 4.21, note that the windowing has considerably reduced the spurious peaks in the spectra but has also further smoothed the peaks. Thus the peak at 0.8π is recognizable with high confidence, but the separation of two close peaks is not so clear for $L=64$. However, the $L=128$ case provides the best balance between separation and detection. On comparing the Blackman-Tukey (Figure 4.18) and Welch estimates, we observe that the results are comparable in terms of variance reduction and smoothing aspects.

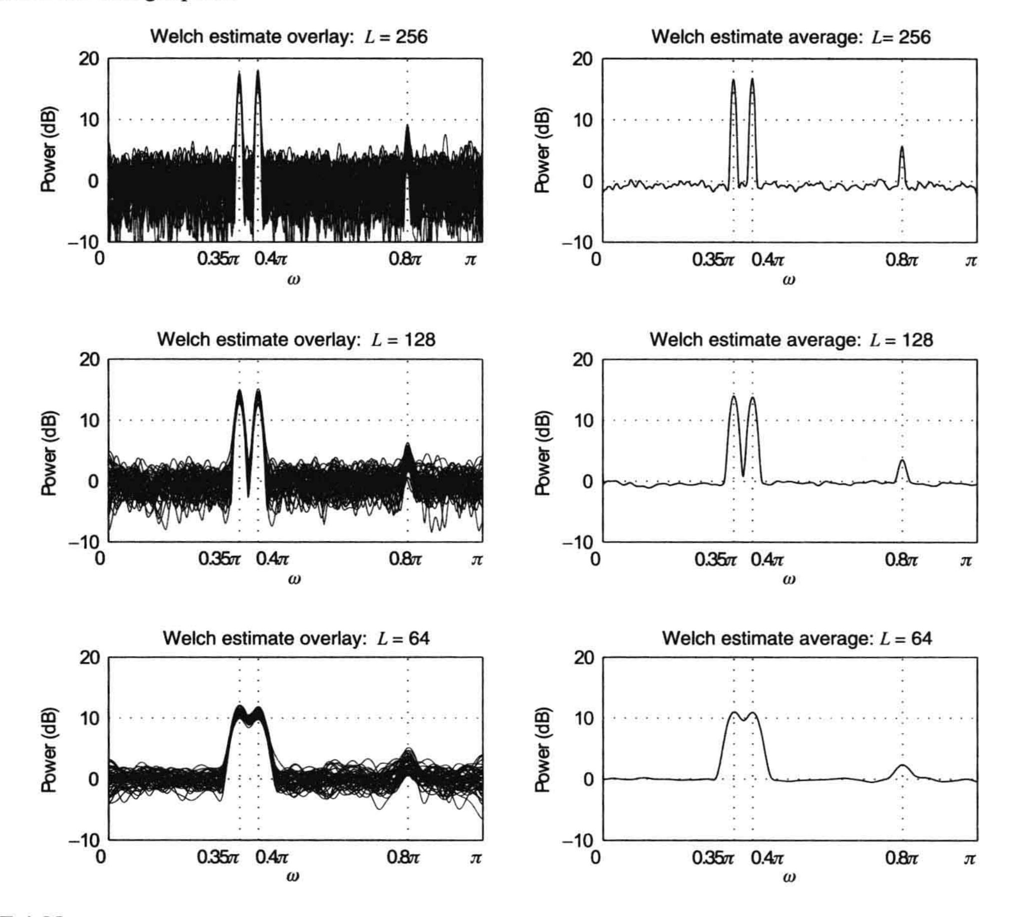


FIGURE 4.22
Estimation of three sinusoids in white noise using Welch's method in Example 4.3.8.

4.3.4 Some Practical Considerations and Examples

The periodogram and its modified version, which is the basic tool involved in the estimation of the power spectrum of stationary signals, can be computed either *directly* from the signal samples $\{x(n)\}_0^{N-1}$ using the DTFT formula

$$\hat{R}_x(e^{j\omega}) = \frac{1}{N}\left|\sum_{n=0}^{N-1} w(n)x(n)e^{-j\omega n}\right|^2 \tag{4.3.71}$$

or *indirectly* using the autocorrelation sequence

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-(N-1)}^{N-1} \hat{r}_x(l)e^{-j\omega l} \tag{4.3.72}$$

where $\hat{r}_x(l)$ is the estimated autocorrelation of the windowed segment $\{w(n)x(n)\}_0^{N-1}$. The periodogram $\hat{R}_x(e^{j\omega})$ provides an unacceptable estimate of the power spectrum because

1. it has a bias that depends on the length N and the shape of the data window $w(n)$ and
2. its variance is equal to the true spectrum $R_x(e^{j\omega})$.

Given a data segment of fixed duration N, there is no way to reduce the bias, or equivalently to increase the resolution, because it depends on the length and the shape of the window. However, we can reduce the variance either by averaging the single periodogram of the data (method of Blackman-Tukey) or by averaging multiple periodograms obtained by partitioning the available record into smaller overlapping segments (method of Bartlett-Welch).

The method of Blackman-Tukey is based on the following modification of the indirect periodogram formula

$$\hat{R}_x^{(PS)}(e^{j\omega}) = \sum_{l=-(L-1)}^{L-1} \hat{r}_x(l)w_a(l)e^{-j\omega l} \tag{4.3.73}$$

which basically involves windowing of the estimated autocorrelation sequence with a proper correlation window. Using only the first $L \ll N$ more-reliable values of the autocorrelation sequence reduces the variance of the spectrum estimate by a factor of approximately L/N. However, at the same time, this reduces the resolution from about $1/N$ to about $1/L$. The recommended range for L is between $0.1N$ and $0.2N$.

The method of Bartlett-Welch is based on partitioning the available data record into windowed overlapping segments of length L, computing their periodograms by using the direct formula (4.3.71), and then averaging the resulting periodograms to compute the estimate

$$\hat{R}_x^{(PA)}(e^{j\omega}) = \frac{1}{KL}\left|\sum_{n=0}^{L-1} x_i(n)e^{-j\omega n}\right|^2 \tag{4.3.74}$$

whose resolution is reduced to approximately $1/L$ and whose variance is reduced by a factor of about $1/K$, where K is the number of segments.

The reduction in resolution and variance of the Blackman-Tukey estimate is achieved by "averaging" the values of the spectrum at consecutive frequency bins by windowing the estimated autocorrelation sequence. In the Bartlett-Welch method, the same effect is achieved by averaging the values of multiple shorter periodograms at the same frequency bin. The PSD estimation methods and their properties are summarized in Table 4.3. The multitaper spectrum estimation method given in the last column of Table 4.3 is discussed in Section 4.5.

Table 4.3. Comparison of PSD estimation methods.

	Periodogram $\hat{R}_x(e^{j\omega})$	Single-periodogram smoothing (Blackman-Tukey): $\hat{R}_x^{(PS)}(e^{j\omega})$	Multiple-periodogram averaging (Bartlett-Welch): $\hat{R}_x^{(PA)}(e^{j\omega})$	Multitaper (Thomson): $\hat{R}_x^{(MT)}(e^{j\omega})$
Description of the method	Compute DFT of data record	Compute DFT of windowed autocorrelation estimate (see Figure 4.17)	Split record into K segments and average their modified periodograms (see Figure 4.20)	Window data record using K orthonormal tapers and average their periodograms (see Figure 4.30)
Basic idea	Natural estimator of $R_x(e^{j\omega})$; the error $\lvert r_x(l)-\hat{r}_x(l)\rvert$ is large for large $\lvert l\rvert$	Local smoothing of $\hat{R}_x(e^{j\omega})$ by weighting $\hat{r}_x(l)$ with a lag window $w_a(l)$	Overlap data records to create more segments; window segments to reduce bias; average periodograms to reduce variance	For properly designed orthogonal tapers, periodograms are independent at each frequency. Hence averaging reduces variance
Bias	Severe for small N; negligible for large N	Asymptotically unbiased	Asymptotically unbiased	Negligible for properly designed tapers
Resolution	$\propto 1/N$	$\propto 1/L$, L is maximum lag	$\propto 1/L$	$\propto 1/N$

			L is segment length	
Variance	Unacceptable: about $R_x^2(e^{j\omega})$ for all N	$R_x^2(e^{j\omega}) \times \frac{E_w}{N}$	$\frac{R_x^2(e^{j\omega})}{K}$ K is number of segments	$\frac{R_x^2(e^{j\omega})}{K}$ K is number of tapers

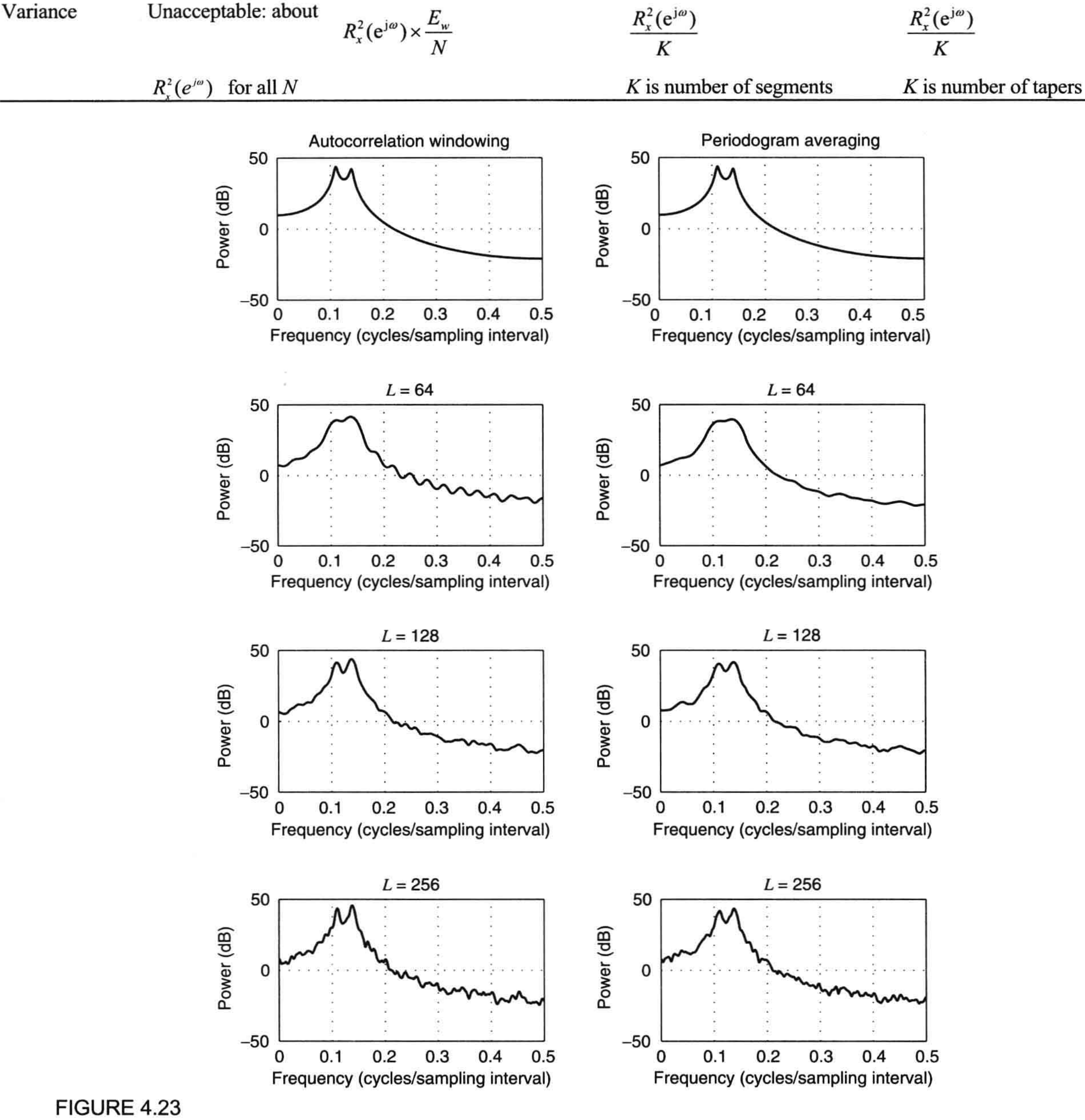


FIGURE 4.23

Illustration of the properties of the power spectrum estimators using autocorrelation windowing (left column) and periodogram averaging (right column) in Example 4.3.9.

EXAMPLE 4.3.9 (COMPARISON OF BLACKMAN-TUKEY AND WELCH-BARTLETT METHODS). Figure 4.23 illustrates the properties of the power spectrum estimators based on autocorrelation windowing and periodogram averaging using the AR(4) model (4.3.24). The top plots show the power spectrum of the process. The left column plots show the power spectrum obtained by windowing the data with a Hanning window and the autocorrelation with a Parzen window of length $L = 64$, 128, and 256. We notice that as the length of the window increases, the resolution decreases and the variance increases. We see a similar behavior with the method of averaged periodograms as the segment length L increases from 64 to 256. Clearly, both methods give comparable results if their parameters are chosen properly.

Example of ocean wave data. To apply spectrum estimation techniques discussed in this chapter to real data, we will use two real-valued time series that are obtained by recording the height of ocean waves as a function of time, as measured by two wave gages of different designs. These two series are shown in Figure 4.24. The top graph shows the wire wave gage data while the bottom graph shows the infrared wave gage data. The frequency responses of these gages are such that—mainly because of its inertia—frequencies higher than 1 Hz cannot be reliably measured. The

frequency range between 0.2 and 1 Hz is also important because the rate at which the spectrum decreases has a physical model associated with it. Both series were collected at a rate of 30 samples per second. There are 4096 samples in each series.[6] We will also use these data to study joint signal analysis in the next section.

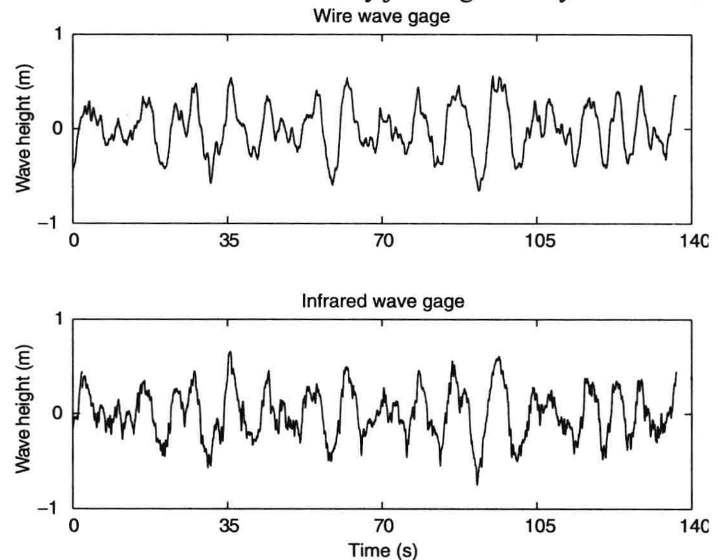


FIGURE 4.24
Display of ocean wave data.

EXAMPLE 4.3.10 (ANALYSIS OF THE OCEAN WAVE DATA)]. Figure 4.25 depicts the periodogram averaging and smoothing estimates of the wire wave gage data. The top row of plots shows the Welch estimate using a Hamming window, $L = 256$, and 50 percent overlap between segments. The bottom row shows the Blackman-Tukey estimate using a Bartlett window and a lag length of $L = 256$. In both cases, a zoomed view of the plots between 0 and 1 Hz is shown in the right column to obtain a better view of the spectra. Both spectral estimates provide a similar spectral behavior, especially over the frequency range of 0 to 1 Hz. Furthermore, both show a broad, low-frequency peak at 0.13 Hz, corresponding to a period of about 8 s. The dominant features of the time series thus can be attributed to this peak and other features in the 0- to 0.2-Hz range. The shape of the spectrum between 0.2 and 1 Hz is a decaying exponential and is consistent with the physical model. Similar results were obtained for the infrared wave gauge data.

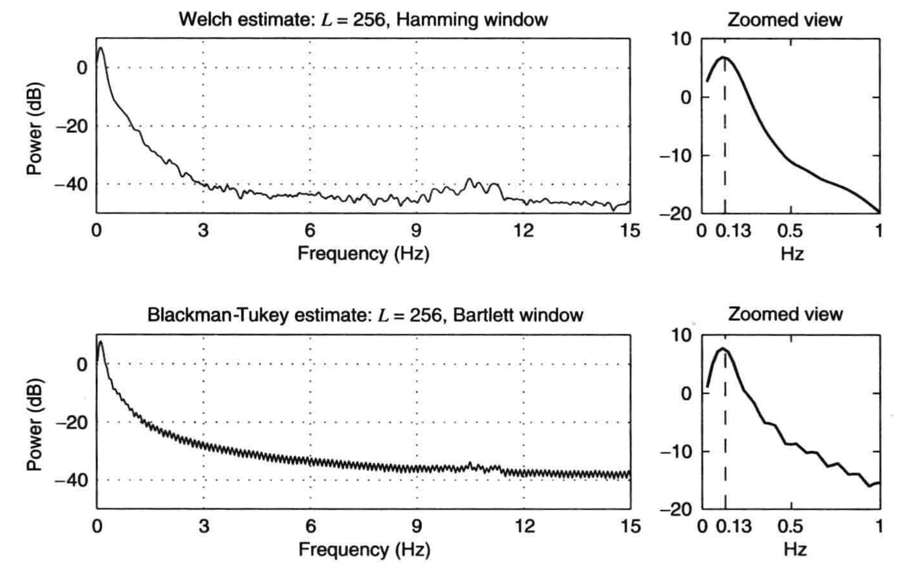


FIGURE 4.25
Spectrum estimation of the ocean wave data using the Welch and Blackman-Tukey methods.

[6] These data were collected by A. Jessup, Applied Physics Laboratory, University of Washington. It was obtained from StatLib, a statistical archive maintained by Carnegie Mellon University.

4.4 Multitaper Power Spectrum Estimation

Tapering is another name for the data windowing operation in the time domain. The periodogram estimate of the power spectrum, discussed in Section 4.3, is an operation on a data record $\{x(n)\}_{n=0}^{N-1}$. One interpretation of this finite-duration data record is that it is obtained by truncating an infinite-duration process $x(n)$ with a rectangular window (or taper). Since bias and variance properties of the periodogram estimate are unacceptable, methods for bias and variance reduction were developed either by smoothing estimates in the frequency domain (using lag windows) or by averaging periodograms computed over several short segments (data windows). Since these window functions (other than the rectangular one) typically taper the response toward both ends of the data record, windows are also referred to as *tapers*.

In 1982, Thomson suggested an alternate approach for producing a “direct” (or “raw” periodogram-based) spectral estimator. In this method, rather than use a single rectangular data taper as in the periodogram estimate, several data tapers are used on the same data record to compute several modified periodograms. These modified periodograms are then averaged (with or without weighting) to produce the multitaper spectral estimate. The central premise of this multitaper approach is that if the data tapers are properly designed orthogonal functions, then, under mild conditions, the spectral estimates would be independent of each other at every frequency. Thus, averaging would reduce the variance while proper design of full-length windows would reduce bias and loss of resolution. Thomson suggested windows based on discrete prolate spheroidal sequences (DPSSs) that form an orthonormal set, although any other orthogonal set with desirable properties can also be used. This DPSS set is also known as the set of *Slepian* tapers. The multitaper method is different in spirit from the other methods in that it does not seek to produce highly smoothed spectra. Detailed discussions of the multitaper approach are given in Thomson (1982) and in Percival and Walden (1993). In this section, we provide a brief sketch of the algorithm.

Estimation of Auto Power Spectrum

Given a data record $\{x(n)\}_{n=0}^{N-1}$ of length N consider a set of K data tapers $\{w_k(n); 0 \le n \le N-1, 0 \le k \le K-1\}$. These tapers are assumed to be orthonormal, that is,

$$\sum_{n=0}^{N-1} w_k(n) w_l(n) = \begin{cases} 1 & k=l \\ 0 & k \ne l \end{cases} \tag{4.4.1}$$

Let $\hat{R}_{k,x}(e^{j\omega})$ be the periodogram estimator based on kth taper. Then, similar to (4.3.2), we obtain

$$\hat{R}_{k,x}(e^{j\omega}) = \frac{1}{N}\left|\sum_{n=0}^{N-1} w_k(n) x(n) e^{-j\omega n}\right|^2 \tag{4.4.2}$$

The simple averaged multitaper (MT) estimator is then defined by

$$\hat{R}_x^{(\mathrm{MT})}(e^{j\omega}) = \frac{1}{K}\sum_{k=0}^{K-1} \hat{R}_{k,x}(e^{j\omega}) \tag{4.4.3}$$

A pictorial description of this multitaper algorithm is shown in Figure 4.26. Another approach, suggested by Thomson, is to apply adaptive weights (both frequency- and data-dependent) prior to averaging to protect against the biasing degradations of different tapers.

In either case, the multitaper estimator is an average of direct spectral estimators (called eigenspectra by Thomson) employing an orthonormal set of tapers. Thomson (1982) showed that under mild conditions, the orthonormality of the tapers results in an approximate independence of each individual $\hat{R}_{k,x}(e^{j\omega})$ at every frequency ω. This approximate independence further implies that the equivalent degrees of freedom for $\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})$ are equal to twice the number of data tapers. This increase in degrees of freedom is enough to shrink the width of the 95 percent confidence interval for $\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})$ and to reduce the variability to the point at which the overall shape of the spectrum is easily recognizable even though the spectrum is not highly smoothed.

Clearly, the success of this approach lies in the selection of K orthonormal tapers. To understand the rationale behind the selection of these tapers, consider the bias or mean of $\hat{R}_{k,x}(e^{j\omega})$. Following (4.3.15), we obtain

$$E\{\hat{R}_{k,x}(e^{j\omega})\} = \frac{1}{2\pi N}\int_{-\pi}^{\pi} R_x(e^{j\theta}) R_{k,w}(e^{j(\omega-\theta)})\,d\theta \tag{4.4.4}$$

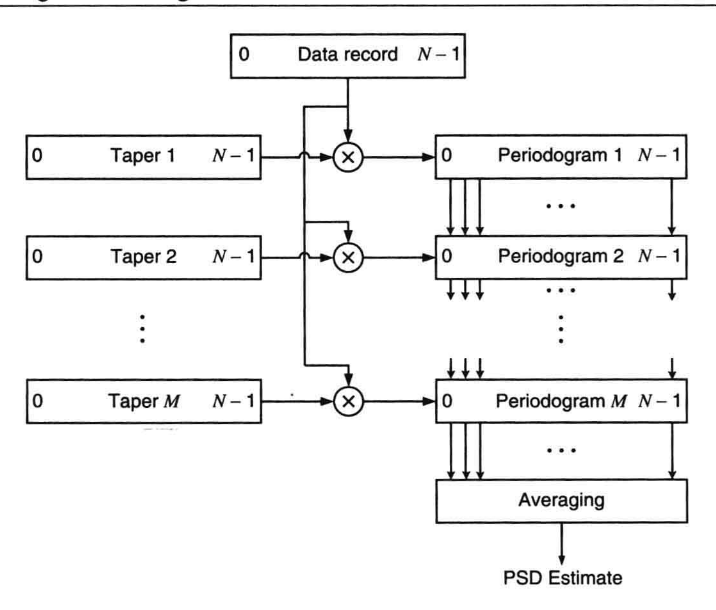


FIGURE 4.26
A pictorial description of the multitaper approach to power spectrum estimation.

where
$$R_{k,w}(e^{j\omega}) = \mathcal{F}\{w_k(n) * w_k(-n)\} = |W_k(e^{j\omega})|^2 \tag{4.4.5}$$

It follows, then, from (4.5.3) that
$$E\{\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})\} = \frac{1}{2\pi N}\int_{-\pi}^{\pi} R_x(e^{j\theta})\bar{R}_w(e^{j(\omega-\theta)})\,d\theta \tag{4.4.6}$$

where
$$\bar{R}_w(e^{j\omega}) \triangleq \frac{1}{K}\sum_{k=0}^{K-1}|W_k(e^{j\omega})|^2 \tag{4.4.7}$$

The function $\bar{R}_w(e^{j\omega})$ is the spectral window of the averaged multitaper estimator, which is obtained by averaging spectra of the individual tapers. Hence, for $\bar{R}_w(e^{j\omega})$ to produce a good leakage-free estimate $\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})$, all K spectral windows must provide good protection against leakage. Therefore, each taper must have low sidelobe levels. Furthermore, the averaging of K individual periodograms also reduces the overall variance of $\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})$. The reduction in variance is possible if the $\hat{R}_{k,x}(e^{j\omega})$ are pairwise uncorrelated with common variance, in which case the variance reduces by a factor of $1/K$.

Thus, we need K orthonormal data tapers such that each one provides a good protection against leakage and such that the resulting individual spectral estimates are nearly uncorrelated. One such set is obtained by using DPSS with parameter W and of orders $k = 0, \cdots, K-1$, where K is chosen to be less than or equal to the number $2W$ (called the *Shannon number*, which is also a fixed-resolution bandwidth). The design of these sequences is discussed in detail in Thomson (1982) and in Percival and Walden (1993). In MATLAB these tapers are generated by using the `[w]=dpss(L,W)` function, where `L` is the length of $2W$ tapers computed in matrix `w`.

The first four 21-point DPSS tapers with $W = 4$ and their Fourier transforms are shown in Figure 4.27 while the next four DPSS tapers are shown in Figure 4.28. It can be seen that higher-order tapers assume both positive and negative values. The zeroth-order taper (like other windows) heavily attenuates data values near $n = 0$ and $n = L$. The higher-order tapers successively give greater weights to these values to the point that tapers for $k \geq K$ have very poor bias properties and hence are not used. This behavior is quite evident in the frequency domain where as the taper order increases, mainlobe width and sidelobe attenuation decrease. The multitapering approach can be interpreted as a technique in which higher-order tapers capture information that is "lost" when only the first taper is used.

In MATLAB the function

[Pxx, Pxxc, F] = PMTM(x, W, Nfft, Fs)

estimates the power spectrum of the data vector `x` in the array `Pxx`, using the multitaper approach. The function uses DPSS tapers with parameter `W` and adaptive weighted averaging as the default method. The 95 percent confidence

interval is available in Pxxc. The size of the DFT used is Nfft, the sampling frequency is Fs, and the frequency values are returned in the vector F.

Another much simpler set of orthonormal tapers was suggested by Reidel and Siderenko (1995). This particular set contains harmonically related sinusoidal tapers. One important aspect of multitapering is to reduce the periodogram variance without reducing resolution caused by smoothing across frequencies. If the spectrum is changing slowly across the band so that sidelobe bias is not severe (recall the argument given for the unbiasedness of the periodogram for the white noise process), then sine tapers can reduce the variance. The kth taper in this set of $k = 0,\ 1,\ \cdots,\ N-1$ tapers is given by

$$w_k(n) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(k+1)(n+1)}{N+1} \qquad n = 0,\ 1,\ \cdots,\ N-1 \tag{4.4.8}$$

where the amplitude term on the right is a normalization factor that ensures orthonormality of the tapers. These sine tapers have much narrower mainlobe but also much higher sidelobes (recall the rectangular window) than the DPSS tapers. Thus they achieve a smaller bias due to smoothing by the mainlobe than the DPSS tapers, but at the expense of sidelobe suppression. Clearly this performance is acceptable if the spectrum is varying slowly. Owing to their simple nature, these tapers can be analyzed analytically, and it can be shown that (Reidel and Siderenko 1995) the k th sinusoidal taper has its spectral energy concentrated in the frequency bands

$$\frac{\pi k}{N+1} \le |\omega| \le \frac{\pi(k+2)}{N+1} \qquad k = 0,\ 1,\ \cdots,\ N-1 \tag{4.4.9}$$

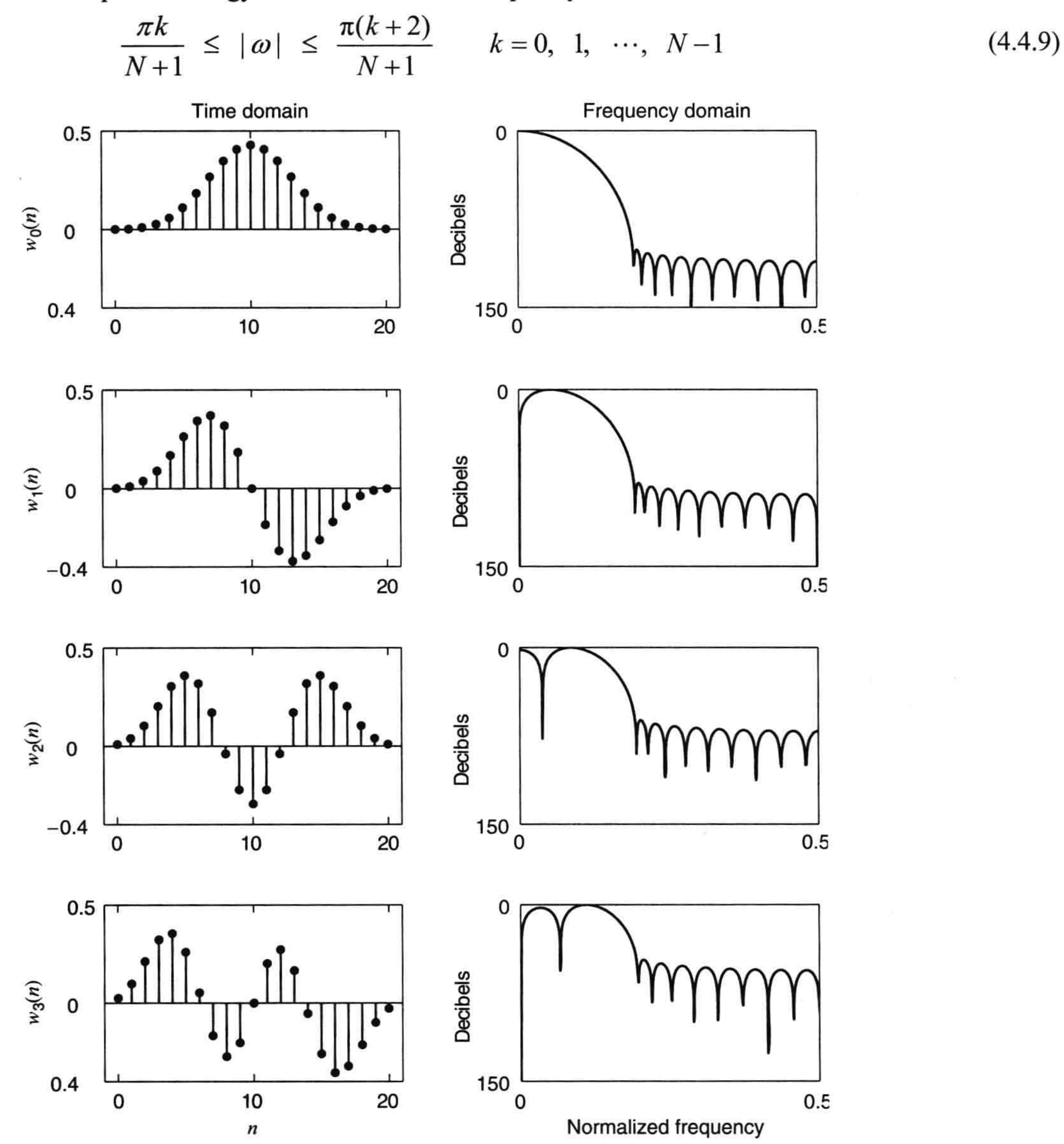


FIGURE 4.27
DPSS data tapers for $k = 0,\ 1,\ 2,\ 3$ in the time and frequency domains.

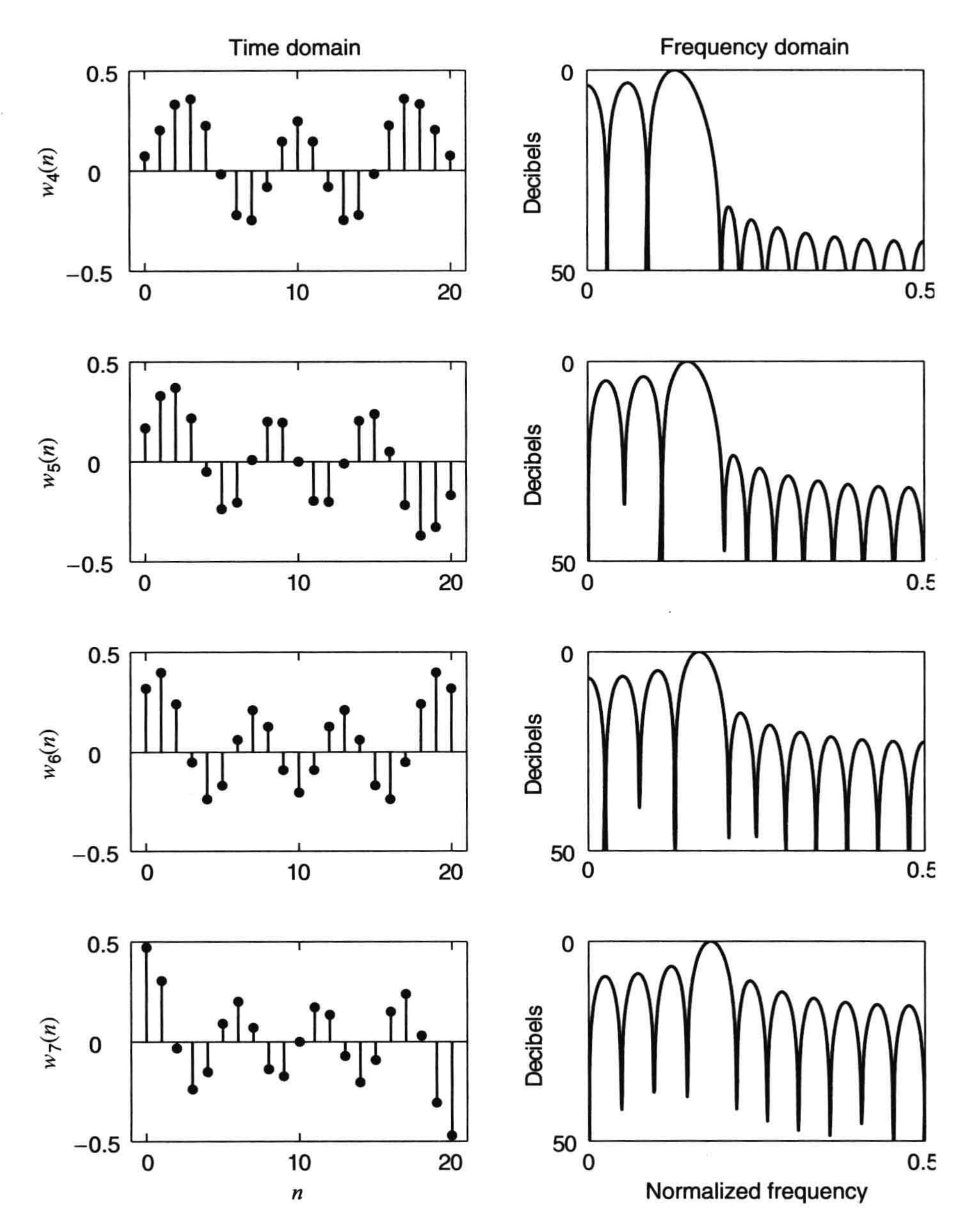


FIGURE 4.28
DPSS data tapers for $k = 4,5,6,7$ in the time and frequency domains.

If the first $K < N$ tapers are used, then the multitaper estimator has the spectral window concentrated in the band

$$\left[-\frac{K+1}{N+1}, \frac{K+1}{N+1}\right] \tag{4.4.10}$$

A summary of the multitaper algorithm performance and its comparison with other PSD estimation methods are given in Table 4.3.

EXAMPLE 4.5.1(THREE SINUSOIDS IN WHITE NOISE). Consider the random process $x(n)$ containing three sinusoids in white noise discussed earlier, that is,

$$x(n) = \cos(0.35\pi n + \varphi_1) + \cos(0.4\pi n + \varphi_2) + 0.25\cos(0.8\pi n + \varphi_3) + v(n)$$

Fifty realizations of $x(n)$, $0 \le n \le N-1$, were processed using the PMTM function to obtain multitaper spectrum estimates for K=3, 5, and 7 Slepian tapers. The results are shown in Figure 4.29 in the form of overlays and averages. Several interesting observations and comparisons with the previous methods can be made. The number of tapers used in the estimation determines the variance and the smearing of the spectrum. When fewer tapers are used, the peaks are sharper and narrower but the noise variance is

larger. After increasing the number of tapers, the variance is decreased but the peaks become wider. When these estimates are compared with those from Welch's method, an interesting feature can be noticed. The broadening of the peaks is not just at the base but is present along the entire length of the peak. Therefore, even with seven tapers, peaks are distinguishable. This feature is due to the bandwidth of the average spectral window due to K tapers.

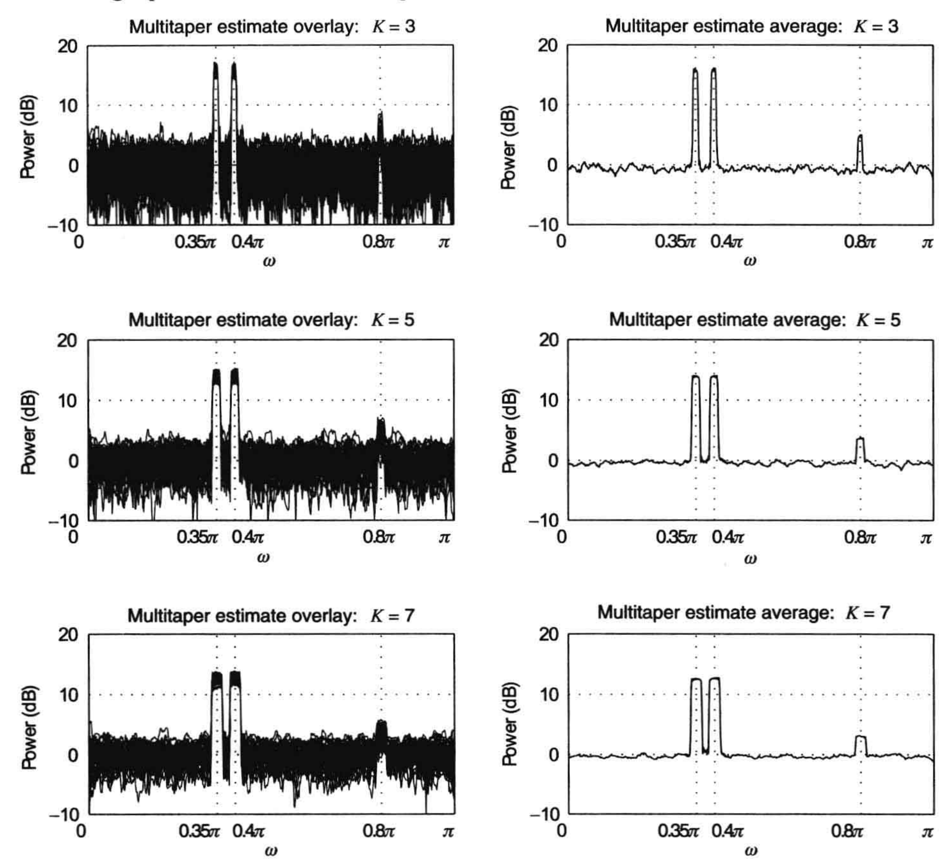


FIGURE 4.29
Spectrum estimation of three sinusoids in white noise using the multitaper method in Example 4.5.1.

EXAMPLE 4.5.2 (OCEAN WAVE DATA). Consider the wire gage wave data of Figure 4.34. The multitaper estimate $\hat{R}_x^{(\mathrm{MT})}(e^{j\omega})$ of these 4096-point data is obtained using the PMTM function in which the parameter W is set to 4. The plots are shown in Figure 4.30. The upper graph shows the spectrum over 0 to 2 Hz while the lower graph shows a zoomed plot over 0 to 0.5 Hz for

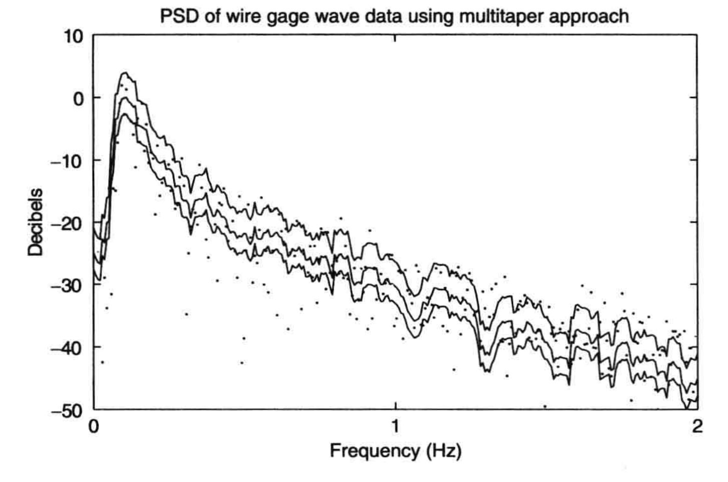


FIGURE 4.30
Spectrum estimation of the wire gage wave data using the multitaper method in Example 4.5.2.

clarity. In each graph, the middle solid is the spectral estimate in decibels while the upper and lower solid curves are the upper and lower limits of the 95 percent confidence interval. For comparison purposes, the "raw" periodogram estimate is also shown as small dots. Clearly, the periodogram has a large variability that is reduced in the multitaper estimate. At the same time, the multitaper estimate is not smooth, but its variability is small enough to follow the shape of the overall structure.

4.5 Summary

In this chapter, we presented many different nonparametric methods for estimating the power spectrum of a wide-sense stationary random process. Nonparametric methods do not depend on any particular model of the process but use estimators that are determined entirely by the data. Therefore, one has to be very careful about the data and the interpretation of results based on them.

We began by revisiting the topic of frequency analysis of deterministic signals. Since the spectrum estimation of random processes is based on the Fourier transformation of data, the purpose of this discussion was to identify and study errors associated with the practical implementation. In this regard, three problems—the sampling of the continuous signal, windowing of the sampled data, and the sampling of the spectrum—were isolated and discussed in detail. Some useful data windows and their characteristics were also given. This background was necessary to understand more complex spectrum estimation methods and their results.

An important topic of autocorrelation estimation was considered next. Although this discussion was not directly related to spectrum estimation, its inclusion was appropriate since one important method (i.e., that of Blackman and Tukey) was based on this estimation. The statistical properties of the estimator and its implementation completed this topic.

The major part of this chapter was devoted to the section on the auto power spectrum estimation. The classical approach was to develop an estimator from the Fourier transform of the given values of the process. This was called the periodogram method, and it resulted in a natural PSD estimator as a Fourier transform of an autocorrelation estimate. Unfortunately, the statistical analysis of the periodogram showed that it was not an unbiased estimator or a consistent estimator; that is, its variability did not decrease with increasing data record length. The modification of the periodogram using the data window lessened the spectral leakage and improved the unbiasedness but did not decrease the variance. Several examples were given to verify these aspects.

To improve the statistical performance of the simple periodogram, we then looked at several possible improvements to the basic technique. Two main directions emerged for reducing the variance: periodogram smoothing and periodogram averaging. These approaches produced consistent and asymptotically unbiased estimates. The periodogram smoothing was obtained by applying the lag window to the autocorrelation estimate and then Fourier-transforming it. This method was due to Blackman and Tukey, and results of its mean and variance were given. The periodogram averaging was done by segmenting the data to obtain several records, followed by windowing to reduce spectral leakage, and finally by averaging their periodograms to reduce variance. This was the well-known Welch-Bartlett method, and the results of its statistical analysis were also given. Finally, implementations based on the DFT and MATLAB were given for both methods along with several examples to illustrate the performance of their estimates. That was based on applying several data windows or tapers to the data followed by averaging of the resulting modified periodograms. The basic principle behind this method was that if the tapers are orthonormal and properly designed (to reduce leakage), then the resulting periodograms can be considered to be independent at each frequency and hence their average would reduce the variance.

Problems

4.1 Let $x_c(t), -\infty < t < \infty$, be a continuous-time signal with Fourier transform $X_c(F)$, $-\infty < F < \infty$, and let $x(n)$ be obtained by sampling $x_c(t)$ every T per sampling interval with its DTFT $X(e^{j\omega})$.

(*a*) Show that the DTFT $X(e^{j\omega})$ is given by

$$X(e^{j\omega}) = F_s \sum_{l=-\infty}^{\infty} X_c(fF_s - lF_s) \qquad \omega = 2\pi f \qquad F_s = \frac{1}{T}$$

(*b*) Let $\tilde{X}_p(k)$ be obtained by sampling $X(e^{j\omega})$ every $2\pi/N$ rad per sampling interval, that is,

$$\tilde{X}_p(k) = X(e^{j2\pi k/N}) = F_s \sum_{l=-\infty}^{\infty} X_c\left(\frac{kF_s}{N} - lF_s\right)$$

Then show that inverse $\text{DFT}(\tilde{X}_p)$ is given by

$$x_p(n) \triangleq \text{IDFT}(\tilde{X}_p) = x_p(n) \triangleq \sum_{m=-\infty}^{\infty} x_c(nT - mNT)$$

4.2 MATLAB provides two functions to generate triangular windows, namely, `bartlett` and `triang`. These two functions actually generate two slightly different coefficients.

(*a*) Use `bartlett` to generate N=11, 31, and 51 length windows $w_B(n)$, and plot their samples, using the `stem` function.

(*b*) Compute the DTFTs $W_B(e^{j\omega})$, and plot their magnitudes over $[-\pi, \pi]$. Determine experimentally the width of the mainlobe as a function of N. Repeat part (*a*) using the `triang` function. How are the lengths and the mainlobe widths different in this case? Which window function is an appropriate one in terms of nonzero samples?

(*c*) Determine the length of the `bartlett` window that has the same mainlobe width as that of a 51-point rectangular window.

4.3 Sidelobes of the window transform contribute to the spectral leakage due to the frequency-domain convolution. One measure of this leakage is the maximum sidelobe height, which generally occurs at the first sidelobe for all windows except the Dolph-Chebyshev window.

(*a*) For simple windows such as the rectangular, Hanning, or Hamming window, the maximum sidelobe height is independent of window length N. Choose N=11, 31, and 51, and determine the maximum sidelobe height in decibels for the above windows.

(*b*) For the Kaiser window, the maximum sidelobe height is controlled by the shape parameter β and is proportional to $\beta / \sinh \beta$. Using several values of β and N, verify the relationship between β and the maximum sidelobe height.

(*c*) Determine the value of β that gives the maximum sidelobe height nearly the same as that of the Hamming window of the same length. Compare the mainlobe widths and the window coefficients of these two windows.

(*d*) For the Dolph-Chebyshev window, all sidelobes have the same height A in decibels. For A=40, 50 and 60 dB, determine the 3-dB mainlobe widths for N=31 length window.

4.4 Let $x(n)$ be given by

$$y(n) = \cos \omega_1 n + \cos(\omega_2 n + \phi) \qquad \text{and} \qquad x(n) = y(n)w(n)$$

where $w(n)$ is a length-N data window. The $|X(e^{j\omega})|^2$ is computed using MATLAB and is plotted over $[0, \pi]$.

(*a*) Let $w(n)$ be a rectangular window. For $\omega_1 = 0.25\pi$ and $\omega_2 = 0.3\pi$, determine the minimum length N so that the two frequencies in the $|X(e^{j\omega})|^2$ plot are barely separable for any arbitrary $\phi \in [-\pi, \pi]$. (You may want to consider the worst possible value of ϕ or experiment, using several values of ϕ.)

(*b*) Repeat part (*a*) for a Hamming window.

(*c*) Repeat part (*a*) for a Blackman window.

4.5 In this problem we will prove that the autocorrelation matrix $\hat{\boldsymbol{R}}_x$ given in (4.2.3), in which the sample correlations are defined by (4.2.1), is a nonnegative definite matrix, that is,

$$\boldsymbol{x}^H \hat{\boldsymbol{R}}_x \boldsymbol{x} \geq 0 \qquad \text{for every } \boldsymbol{x} \geq \boldsymbol{0}$$

(*a*) Show that $\hat{\boldsymbol{R}}_x$ can be decomposed into the product $\mathbf{X}^H\mathbf{X}$, where $\mathbf{X}$ is called a data matrix. Determine the form of $\mathbf{X}$.

(*b*) Using the above decomposition, now prove that $\boldsymbol{x}^H \hat{\boldsymbol{R}}_x \boldsymbol{x} \geq 0$, for every $\boldsymbol{x} \geq \boldsymbol{0}$.

4.6 An alternative autocorrelation estimate $\check{r}_x(l)$ is given in (4.2.13) and is repeated below.

$$\check{r}_x(l) = \begin{cases} \dfrac{1}{N-l} \displaystyle\sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq L < N \\ \check{r}_x^*(-l) & -N < -L \leq l < 0 \\ 0 & \text{elsewhere} \end{cases}$$

(*a*) Show that the mean of $\check{r}_x(l)$ is equal to $r_x(l)$ and an approximate expression for the variance of $\check{r}_x(l)$.

(*b*) Show that the mean of the corresponding periodogram [that is, $\check{R}_x(e^{j\omega}) \triangleq \mathcal{F}[\check{r}_x(l)]$] is given by

$$E\{\check{R}_x(e^{j\omega})\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\phi}) W_R(e^{j\omega-\phi}) d\varphi$$

where $W_R(e^{j\omega})$ is the DTFT of the rectangular window and is sometimes called the *Dirichlet kernel*.

4.7 Consider the above unbiased autocorrelation estimator $\check{r}_x(l)$ of a zero-mean white Gaussian process with variance σ_x^2.

(*a*) Determine the variance of $\check{r}_x(l)$. Compute its limiting value as $l \to \infty$.

(*b*) Repeat part (*a*) for the biased estimator $\hat{r}_x(l)$. Comment on any differences in the results.

4.8 Show that the autocorrelation matrix $\check{\boldsymbol{R}}_x$ formed by using $\check{r}_x(l)$ is not nonnegative definite, that is,

$$\boldsymbol{x}^H \hat{\boldsymbol{R}}_x \boldsymbol{x} < 0 \qquad \text{for some } \boldsymbol{x} \geq 0$$

4.9 In this problem, we will show that the periodogram $\hat{R}_x(e^{j\omega})$ can also be expressed as a DTFT of the autocorrelation estimate

$\hat{r}_x(l)$ given in (4.2.1).

(*a*) Let $v(n) = x(n)w_R(n)$, where $w_R(n)$ is a rectangular window of length N. Show that

$$\hat{r}_x(l) = \frac{1}{N} v(l) * v^*(-l) \tag{P.1}$$

(*b*) Take the DTFT of (P.1) to show that

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-N+1}^{N-1} \hat{r}_x(l) e^{-j\omega l}$$

4.10 Consider the following simple windows over $0 \le n \le N-1$: rectangular, Bartlett, Hanning, and Hamming.

(*a*) Determine analytically the DTFT of each of the above windows.

(*b*) Sketch the magnitude of these Fourier transforms for $N = 31$.

(*c*) Verify your sketches by performing a numerical computation of the DTFT using MATLAB.

4.11 The Parzen window is given by

$$w_P(l) = \begin{cases} 1-6\left(\frac{l}{L}\right)^2 + 6\left(\frac{l}{L}\right)^3 & 0 \le |l| \le \frac{L}{2} \\ 2\left(1-\frac{l}{L}\right)^3 & \frac{L}{2} < |l| < L \\ 0 & \text{elsewhere} \end{cases} \tag{P.2}$$

(*a*) Show that its DTFT is given by

$$W_P(e^{j\omega}) \simeq \left[\frac{\sin(\omega L/4)}{\sin(\omega/4)}\right]^4 \ge 0 \tag{P.3}$$

Hence using the Parzen window as a correlation window always produces nonnegative spectrum estimates.

(*b*) Using MATLAB, compute and plot the time-domain window $w_P(l)$ and its frequency-domain response $W_P(e^{j\omega})$ for L=5, 10, and 20.

(*c*) From the frequency-domain plots in part (*b*) experimentally determine the 3-dB mainlobe width $\Delta\omega$ as a function of L.

4.12 The variance reduction ratio of a correlation window $w_a(l)$ is defined as

$$\frac{\text{var}\{\hat{R}_x^{(PS)}(e^{j\omega})\}}{\text{var}\{\hat{R}_x(e^{j\omega})\}} \simeq E_w / N \qquad 0 < \omega < \pi$$

where

$$E_w = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_a^2(e^{j\omega}) d\omega = \sum_{l=-(L-1)}^{L-1} w_a^2(l)$$

(*a*) Using MATLAB, compute and plot E_w as a function of L for the following windows: rectangular, Bartlett, Hanning, Hamming, and Parzen.

(*b*) Using your computations above, show that for $L \gg 1$, the variance reduction ratio for each window is given by the formula in the following table.

Window name	Variance reduction factor
Rectangular	$2L/N$
Bartlett	$0.667L/N$
Hanning	$0.75L/N$
Hamming	$0.7948L/N$
Parzen	$0.539L/N$

4.13 For $L > 100$, the direct computation of $\hat{r}_x(l)$ using (4.3.39) is time-consuming; hence an indirect computation using the DFT can be more efficient. This computation is implemented by the following steps:

- Given the sequence $\{x(n)\}_{n=0}^{N-1}$, pad enough zeros to make it a $(2N-1)$-point sequence.
- Compute the N_{FFT}-point FFT of $x(n)$ to obtain $\tilde{X}(k)$, where N_{FFT} is equal to the next power-of-2 number that is greater than or equal to $2N-1$.
- Compute $1/N\,|\tilde{X}(k)|^2$ to obtain $\tilde{\hat{R}}(k)$.
- Compute the N_{FFT}-point IFFT of $\tilde{\hat{R}}(k)$ to obtain $\hat{r}_x(l)$.

Develop a MATLAB function `rx = autocfft(x,L)` which computes $\hat{r}_x(l)$, over $-L \le l \le L$. Compare this function with the `autoc` function discussed in the chapter in terms of the execution time for $L \ge 100$.

4.14 The Welch-Bartlett estimate $\hat{R}_x^{(\text{PA})}(k)$ is given by

$$\hat{R}_x^{(\text{PA})}(k) = \frac{1}{KL}\sum_{i=0}^{K-1} |X_i(e^{j\omega})|^2$$

If $x(n)$ is real-valued, then the sum in the above expression can be evaluated more efficiently. Let K be an even number. Then we will combine two real-valued sequences into one complex-valued sequence and compute one FFT, which will reduce the overall computations. Specifically, let

$$g_r(n) \triangleq x_{2r}(n) + jx_{2r+1}(n) \qquad n = 0,\ 1,\ \cdots,\ L-1,\ r = 0,\ 1,\ \cdots,\ \frac{K}{2}-1$$

Then the L-point DFT of $g_r(n)$ is given by

$$\tilde{G}_r(k) = \tilde{X}_{2r}(k) + j\tilde{X}_{2r+1}(k) \qquad k = 0,\ 1,\ \cdots,\ L-1,\ r = 0,\ 1,\ \cdots,\ \frac{K}{2}-1$$

(*a*) Show that

$$|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2 = 2[|\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2] \qquad k,\ r = 0,\ \cdots,\ \frac{K}{2}-1$$

(*b*) Determine the resulting expression for $\hat{R}_x^{(\text{PA})}(k)$ in terms of $\tilde{G}(k)$.

(*c*) What changes are necessary if K is an odd number? Provide detailed steps for this case.

4.15 Since $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is a PSD estimate, one can determine autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$ from Welch's method as

$$\hat{r}_x^{(\text{PA})}(l) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{R}_x^{(\text{PA})}(e^{j\omega})e^{j\omega l}\, d\omega \tag{P.4}$$

Let $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ be the samples of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ according to

$$\tilde{\hat{R}}_x^{(\text{PA})}(k) \triangleq \hat{R}_x^{(\text{PA})}(e^{j2\pi k/N_{\text{FFT}}}) \quad 0 \le k \le N_{\text{FFT}}-1$$

(*a*) Show that the IDFT $\tilde{\hat{r}}_x^{(\text{PA})}(l)$ of $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ is an aliased version of the autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$.

(*b*) If the length of the overlapping data segment in Welch's method is L, how should N_{FFT} be chosen to avoid aliasing in $\tilde{\hat{r}}_x^{(\text{PA})}(l)$?

4.16 Show that the coherence function $\mathcal{G}_{xy}^2(\omega)$ is invariant under linear transformation, that is, if $x_1(n) = h_1(n) * x(n)$ and $y_1(n) = h_2(n) * y(n)$, then

$$\mathcal{G}_{xy}^2(\omega) = \mathcal{G}_{x_1y_2}^2(\omega)$$

4.17 Bartlett's method is a special case of Welch's method in which nonoverlapping sections of length L are used without windowing in the periodogram averaging operation.

(*a*) Show that the ith periodogram in this method can be expressed as

$$\hat{R}_{x,i}(e^{j\omega}) = \sum_{l=-L}^{L} \hat{r}_{x,i}(l)w_B(l)e^{-j\omega l} \tag{P.5}$$

where $w_B(l)$ is a $(2L-1)$-length Bartlett window.

(*b*) Let $\mathbf{u}(e^{j\omega}) \triangleq [1 \ \ e^{j\omega} \ \ \cdots \ \ e^{j(M-1)\omega}]^T$. Show that $\hat{R}_{x,i}(e^{j\omega})$ in (P.5) can be expressed as a quadratic product

$$\hat{R}_{x,i}(e^{j\omega}) = \boldsymbol{u}^H(e^{j\omega})\hat{\boldsymbol{R}}_{x,i}\boldsymbol{u}(e^{\omega}) \tag{P.6}$$

where $\hat{\boldsymbol{R}}_{x,i}$ is the autocorrelation matrix of $\hat{r}_{x,i}(l)$ values.

(*c*) Finally, show that the Bartlett estimate is given by

$$\hat{R}_x^{(B)}(e^{j\omega}) = \frac{1}{K}\sum_{i=1}^{K} \boldsymbol{u}^H(e^{j\omega})\hat{\boldsymbol{R}}_{x,i}\boldsymbol{u}(e^{\omega}) \tag{P.7}$$

4.18 In this problem, we will explore a spectral estimation technique that uses combined data and correlation weighting (Carter and Nuttall 1980). In this technique, the following steps are performed:

- Given $\{x(n)\}_{n=0}^{N-1}$, compute the Welch-Bartlett estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ by choosing the appropriate values of L and D.
- Compute the autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$, $-L \le l \le L$, using the approach described in Problem 4.15.
- Window $\hat{r}_x^{(\text{PA})}(l)$, using a lag window $w_a(l)$ to obtain $\hat{r}_x^{(\text{CN})}(l) \triangleq \hat{r}_x^{(\text{PA})}(l)w_a(n)$.
- Finally, compute the DTFT of $\hat{r}_x^{(\text{CN})}(n)$ to obtain the new spectrum estimate $\hat{R}_x^{(\text{CN})}(e^{j\omega})$.

(*a*) Determine the bias of $\hat{R}_x^{(\mathrm{CN})}(\mathrm{e}^{\mathrm{j}\omega})$.

(*b*) Comment on the effect of additional windowing on the variance and resolution of the estimate.

(*c*) Implement this technique in MATLAB, and compute spectral estimates of the process containing three sinusoids in white noise, which was discussed in the chapter. Experiment with various values of *L* and with different windows. Compare your results to those given for the Welch-Bartlett and Blackman-Tukey methods.

4.19 Explain why we use the scaling factor

$$\sum_{n=0}^{L-1} w^2(n)$$

which is the energy of the data window in the Welch-Bartlett method.

4.20 Consider the basic periodogram estimator $\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega})$ at the zero frequency, that is, at $\omega = 0$.

(*a*) Show that

$$\hat{R}_x(\mathrm{e}^{\mathrm{j}0}) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)\mathrm{e}^{\mathrm{j}0}\right|^2 = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)\right|^2$$

(*b*) If $x(n)$ is a real-valued white Gaussian process with variance σ_x^2, determine the mean and variance of $\hat{R}_x(\mathrm{e}^{\mathrm{j}0})$.

(*c*) Determine if $\hat{R}_x(\mathrm{e}^{\mathrm{j}0})$ is a consistent estimator by evaluating the variance as $N \to \infty$.

4.21 Consider Bartlett's method for estimating $R_x(\mathrm{e}^{\mathrm{j}0})$ using $L = 1$; that is, we use nonoverlapping segments of single samples. The periodogram of one sample $x(n)$ is simply $|x(n)|^2$. Thus we have

$$\hat{R}_x^{(\mathrm{B})}(\mathrm{e}^{\mathrm{j}0}) = \frac{1}{N}\sum_{n=0}^{N-1} \hat{R}_{x,n}(\mathrm{e}^{\mathrm{j}0}) = \frac{1}{N}\sum_{n=0}^{N-1} |x(n)|^2$$

Again assume that $x(n)$ is a real-valued white Gaussian process with variance σ_x^2.

(*a*) Determine the mean and variance of $\hat{R}_x^{(\mathrm{B})}(\mathrm{e}^{\mathrm{j}0})$.

(*b*) Compare the above result with those in Problem 4.20. Comment on any differences.

4.22 One desirable property of lag or correlation windows is that their Fourier transforms are nonnegative.

(*a*) Formulate a procedure to generate a symmetric lag window of length 2*L*+1 with nonnegative Fourier transform.

(*b*) Using the Hanning window as a prototype in the above procedure, determine and plot a 31-length lag window. Also plot its Fourier transform.

4.23 Consider the following random process

$$x(n) = \sum_{k=1}^{4} A_k \sin(\omega_k n + \phi_k) + v(n)$$

where

$A_1 = 1$	$A_2 = 0.5$	$A_3 = 0.5$	$A_4 = 0.25$
$\omega_1 = 0.1\pi$	$\omega_2 = 0.6\pi$	$\omega_3 = 0.65\pi$	$\omega_4 = 0.8\pi$

and the phases $\{\phi_i\}_{i=1}^{4}$ are IID random variables uniformly distributed over $[-\pi, \pi]$. Generate 50 realizations of $x(n)$ for $0 \le n \le 256$.

(*a*) Compute the Blackman-Tukey estimates for *L*=32, 64, and 128, using the Bartlett lag window. Plot your results, using overlay and averaged estimates. Comment on your plots.

(*b*) Repeat part (*a*), using the Parzen window.

(*c*) Provide a qualitative comparison between the above two sets of plots.

4.24 Consider the random process given in Problem 4.23.

(*a*) Compute the Bartlett estimate, using *L*=16, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(*b*) Compute the Welch estimate, using 50 percent overlap, Hamming window, and *L*=16, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(*c*) Provide a qualitative comparison between the above two sets of plots.

4.25 Consider the random process given in Problem 4.23.

(*a*) Compute the multitaper spectrum estimate, using *K*=3, 5, and 7 Slepian tapers. Plot your results, using overlay and averaged estimates. Comment on your plots.

(*b*) Make a qualitative comparison between the above plots and those obtained in Problems 4.23 and 4.24.

4.26 Generate 1000 samples of an AR(1) process using $a = -0.9$. Determine its theoretical PSD.

(*a*) Determine and plot the periodogram of the process along with the true spectrum. Comment on the plots.

(*b*) Compute the Blackman-Tukey estimates for *L*=10, 20, 50, and 100. Plot these estimates along with the true spectrum. Comment

on your results.

(*c*) Compute the Welch estimates for 50 percent overlap, Hamming window, and L=10, 20, 50, and 100. Plot these estimates along with the true spectrum. Comment on your results.

4.27 Generate 1000 samples of an AR(1) process using $a = 0.9$. Determine its theoretical PSD.

(*a*) Determine and plot the periodogram of the process along with the true spectrum. Comment on the plots.

(*b*) Compute the Blackman-Tukey estimates for L=10, 20, 50, and 100 Plot these estimates along with the true spectrum. Comment on your results.

(*c*) Compute the Welch estimates for 50 percent overlap, Hamming window, and L=10, 20, 50, and 100. Plot these estimates along with the true spectrum. Comment on your results.

4.28 Multitaper estimation technique requires a properly designed orthonormal set of tapers for the desired performance. One set discussed in the chapter was that of harmonically related sinusoids given in (4.5.8).

(*a*) Design a MATLAB function `[tapers] = sine_tapers(N,K)` that generates $K < N$ sinusoidal tapers of length N.

(*b*) Using the above function, compute and plot the Fourier transform magnitudes of the first 5 tapers of length 51.

4.29 Design a MATLAB function `Pxx = psd_sinetaper(x,K)` that determines the multitaper estimates using the sine tapers.

(*a*) Apply the function `psd_sinetaper` to the AR(1) process given in Problem 4.26, and compare its performance.

(*b*) Apply the function `psd_sinetaper` to the AR(1) process given in Problem 4.27, and compare its performance.

CHAPTER 5

Optimum Linear Filters

In this chapter, we present the theory and application of optimum linear filters and predictors. We concentrate on linear filters that are optimum in the sense of minimizing the mean square error (MSE). The minimum MSE (MMSE) criterion leads to a theory of linear filtering that is elegant and simple, involves only second-order statistics, and is useful in many practical applications. The optimum filter designed for a given set of second-order moments can be used for any realizations of stochastic processes with the same moments.

We start with the general theory of linear MMSE estimators and their computation. Then we apply the general theory to the design of optimum FIR filters and linear predictors for both nonstationary and stationary processes (Wiener filters). We continue with the design of nonparametric (impulse response) and parametric (pole-zero) optimum IIR filters and predictors for stationary processes. Then we present the design of optimum filters for inverse system modeling, blind deconvolution, and their application to equalization of data communication channels. We conclude with a concise introduction to optimum matched filters and eigenfilters that maximize the output SNR. These signal processing methods find extensive applications in digital communication, radar, and sonar systems.

5.1 Optimum Signal Estimation

As we discussed in Chapter 1, the solution of many problems of practical interest depends on the ability to accurately estimate the value $y(n)$ of a signal (*desired response*) by using a set of values (*observations* or *data*) from another related signal or signals. Successful estimation is possible if there is significant statistical dependence or correlation between the signals involved in the particular application. For example, in the linear prediction problem we use the M past samples $x(n-1), x(n-2), \cdots, x(n-M)$ of a signal to estimate the current sample $x(n)$. The echo canceler in Figure 1.16 uses the transmitted signal to form a replica of the received echo. Although the signals in these and other similar applications have different physical origins, the mathematical formulations of the underlying signal processing problems are very similar.

In array signal processing, the data are obtained by using M different sensors. The situation is simpler for filtering applications, because the data are obtained by delaying a single discrete-time signal; that is, we have $x_k(n) = x(n+1-k),\ 1 \le k \le M$ (see Figure 5.1). Further simplifications are possible in linear prediction, where both the desired response and the data are time samples of the same signal, for example, $y(n) = x(n)$ and $x_k(n) = x(n-k),\ 1 \le k \le M$. As a result, the design and implementation of optimum filters and predictors are simpler than those for an optimum array processor.

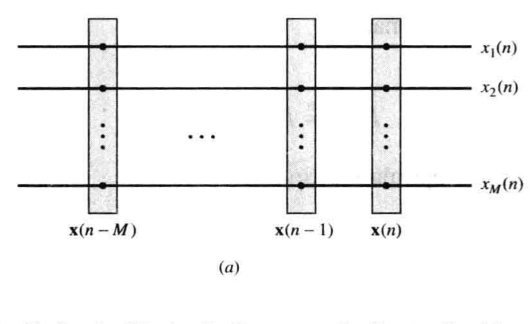
Since array processing problems are the most general ones, we will formulate and solve the following estimation problem: Given a set of data $x_k(n)$ for $1 \le k \le M$, determine an estimate $\hat{y}(n)$, of the desired response $y(n)$, using the rule (*estimator*)

$$\hat{y}(n) \triangleq H\{x_k(n),\ 1 \le k \le M\} \tag{5.1.1}$$

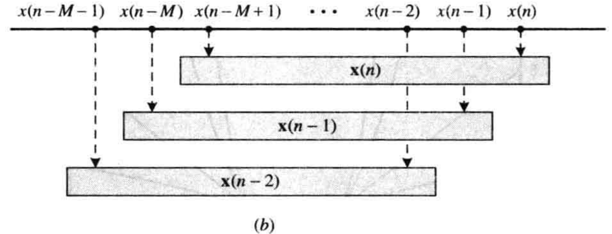
which, in general, is a nonlinear function of the data. When $x_k(n) = x(n+1-k)$, the estimator takes on the form of a discrete-time filter that can be linear or nonlinear, time-invariant or time-varying, and with a finite- or infinite-duration impulse response. Linear filters can be implemented using any direct, parallel, cascade, or lattice-ladder structure.

The difference between the estimated response $\hat{y}(n)$ and the desired response $y(n)$, that is,

$$e(n) \triangleq y(n) - \hat{y}(n) \tag{5.1.2}$$



(a)



(b)

FIGURE5.1
Illustration of the data vectors for (*a*) array processing (multiple sensors) and (*b*) FIR filtering or prediction (single sensor) applications.

is known as the *error signal*. We want to find an estimator whose output approximates the desired response as closely as possible according to a certain performance criterion. We use the term *optimum estimator* or *optimum signal processor* to refer to such an estimator. We stress that *optimum* is not used as a synonym for *best*; it simply means the best under the given set of assumptions and conditions. If either the criterion of performance or the assumptions about the statistics of the processed signals change, the corresponding optimum filter will change as well. Therefore, an optimum estimator designed for a certain performance metric and set of assumptions may perform poorly according to some other criterion or if the actual statistics of the processed signals differ from the ones used in the design. For this reason, the sensitivity of the performance to deviations from the assumed statistics is very important in practical applications of optimum estimators.

Therefore, the design of an optimum estimator involves the following steps:

1. Selection of a computational structure with well-defined parameters for the implementation of the estimator.
2. Selection of a criterion of performance or cost function that measures the performance of the estimator under some assumptions about the statistical properties of the signals to be processed.
3. Optimization of the performance criterion to determine the parameters of the optimum estimator.
4. Evaluation of the optimum value of the performance criterion to determine whether the optimum estimator satisfies the design specifications.

Many practical applications (e.g., speech, audio, and image coding) require subjective criteria that are difficult to express mathematically. Thus, we focus on criteria of performance that (1) only depend on the estimation error $e(n)$, (2) provide a sufficient measure of the user satisfaction, and (3) lead to a mathematically tractable problem. We generally select a criterion of performance by compromising between these objectives.

Since, in most applications, negative and positive errors are equally harmful, we should choose a criterion that weights both negative and positive errors equally. Choices that satisfy this requirement include the absolute value of the error $|e(n)|$, or the squared error $|e(n)|^2$, or some other power of $|e(n)|$ (see Figure 5.2). The emphasis put on different values of the error is a key factor when we choose a criterion of performance. For example, the squared-error criterion emphasizes the effect of large errors much more than the absolute error criterion. Thus, the squared-error criterion is more sensitive to outliers (occasional large values) than the absolute error criterion is.

To develop a mathematical theory that will help to design and analyze the performance of optimum estimators, we assume that the desired response and the data are realizations of stochastic processes. Furthermore, although in practice the estimator operates on specific realizations of the input and desired response signals, we wish to design an estimator with good performance across all members of the ensemble, that is, an estimator that "works well on average." Since, at any fixed time n, the quantities $y(n)$, $x_k(n)$ for $1 \le k \le M$, and $e(n)$ are random variables, we should choose a criterion that involves the ensemble or time averaging of some function of $|e(n)|$.

Here is a short list of potential criteria of performance:

1. The mean square error criterion

$$P(n) \triangleq E\{|e(n)|^2\} \tag{5.1.3}$$

which leads, in general, to a nonlinear optimum estimator.

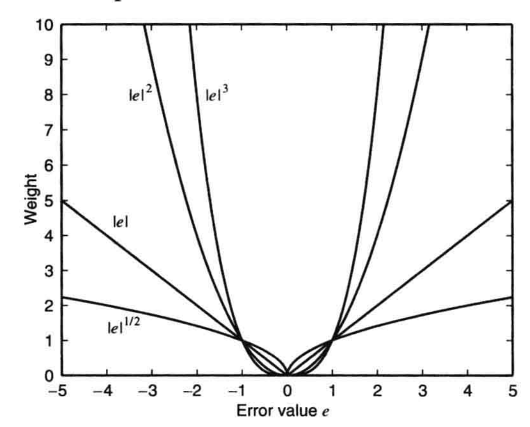


FIGURE 5.2
Graphical illustration of various error-weighting functions.

2. The mean α th-order error criterion $E\{|e(n)|^\alpha\}$, $\alpha \neq 2$. Using a lower- or higher-order moment of the absolute error is more appropriate for certain types of non-Gaussian statistics than the MSE (Stuck 1978).
3. The sum of squared errors (SSE)

$$E(n_i, n_f) \triangleq \sum_{n=n_i}^{n_f} |e(n)|^2 \tag{5.1.4}$$

which, if it is divided by $n_f - n_i + 1$, provides an estimate of the MSE.

The MSE criterion (5.1.3) and the SSE criterion (5.1.4) are the most widely used because they (1) are mathematically tractable, (2) lead to the design of useful systems for practical applications, and (3) can serve as a yardstick for evaluating estimators designed with other criteria (e.g., signal-to-noise ratio, maximum likelihood). In most practical applications, we use linear estimators, which further simplifies their design and evaluation.

Mean square estimation is a rather vast field that was originally developed by Gauss in the nineteenth century. The current theories of estimation and optimum filtering started with the pioneering work of Wiener and Kolmogorov that was later extended by Kalman, Bucy, and others. Some interesting historical reviews are given in Kailath (1974) and Sorenson (1970).

5.2 Linear Mean Square Error Estimation

In this section, we develop the theory of linear MSE estimation. We concentrate on linear estimators for various reasons, including mathematical simplicity and ease of implementation. The problem can be stated as follows:

> Design an estimator that provides an estimate $\hat{y}(n)$ of the desired response $y(n)$ using a *linear* combination of the data $x_k(n)$ for $1 \le k \le M$, such that the MSE $E\{|y(n) - \hat{y}(n)|^2\}$ is minimized.

More specifically, the linear estimator is defined by

$$\hat{y}(n) \triangleq \sum_{k=1}^{M} c_k^*(n) x_k(n) \tag{5.2.1}$$

and the goal is to determine the coefficients $c_k(n)$ for $1 \le k \le M$ such that the MSE (5.1.3)is minimized. In general, a new set of optimum coefficients should be computed for each time instant n. Since we assume that the desired response and the data are realizations of stochastic processes, the quantities $y(n), x_1(n), \ldots, x_M(n)$ are random variables at any fixed time n. For convenience, we formulate and solve the estimation problem at a fixed time instant n. Thus, we drop the time index n and restate the problem as follows:

Estimate a random variable y (the desired response) from a set of related random variables $x_1, x_2, \ldots, x_M$ (data) using the linear estimator

$$\hat{y} \triangleq \sum_{k=1}^{M} c_k^* x_k = \mathbf{c}^{\mathrm{H}}\mathbf{x} \tag{5.2.2}$$

where

$$\mathbf{x}=[x_1\ x_2\ \cdots\ x_M]^{\mathrm{T}} \tag{5.2.3}$$

is the *input data vector* and

$$\mathbf{c}=[c_1\ c_2\ \cdots\ c_M]^{\mathrm{T}} \tag{5.2.4}$$

is the *parameter* or *coefficient vector* of the estimator.

Unless otherwise stated, *all* random variables are assumed to have *zero*-mean values. The number M of data components used is called the *order* of the estimator. The linear estimator (5.2.2) is represented graphically as shown in Figure 5.3 and involves a computational structure known as the *linear combiner*. The MSE

$$P \triangleq E\{|e|^2\} \tag{5.2.5}$$

where

$$e \triangleq y - \hat{y} \tag{5.2.6}$$

is a function of the parameters c_k. Minimization of (5.2.5) with respect to parameters c_k leads to a linear estimator $\mathbf{c}_o$ that is optimun in the MSE sense. The parameter vector $\mathbf{c}_o$ is known as the *linear MMSE (LMMSE) estimator* and $\hat{y}_o$ as the LMMSE estimate.

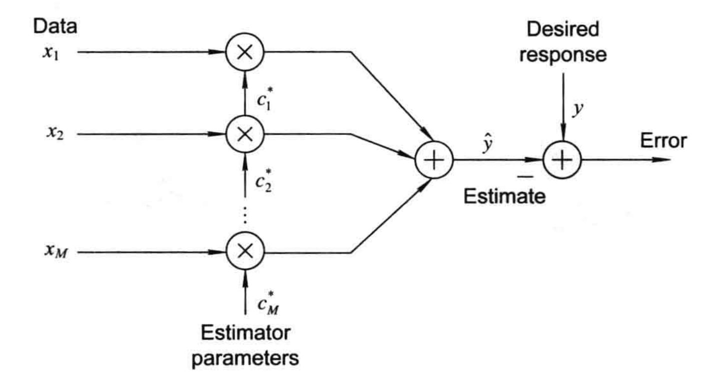


FIGURE 5.3
Block diagram representation of the linear estimator.

5.2.1 *Error Performance Surface*

To determine the linear MMSE estimator, we seek the value of the parameter vector $\mathbf{c}$ that minimizes the function (5.2.5). To this end, we want to express the MSE as a function of the parameter vector $\mathbf{c}$ and to understand the nature of this dependence.

By using (5.2.5), (5.2.6), (5.2.2), and the linearity property of the expectation operator, the MSE is given by

$$\begin{aligned} P(\boldsymbol{c}) &= E\{|e|^2\} = E\{(y-\boldsymbol{c}^{\mathrm{H}}\boldsymbol{x})(y^* - \boldsymbol{x}^{\mathrm{H}}\boldsymbol{c})\} \\ &= E\{|y|^2\} - \boldsymbol{c}^{\mathrm{H}} E\{\boldsymbol{x}y^*\} - E\{y\boldsymbol{x}^{\mathrm{H}}\}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}} E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\}\boldsymbol{c} \end{aligned}$$

or more compactly,

$$P(\boldsymbol{c}) = P_y - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{d} - \boldsymbol{d}^{\mathrm{H}}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}}\boldsymbol{R}\boldsymbol{c} \tag{5.2.7}$$

where
$$P_y \triangleq E\{|y|^2\} \tag{5.2.8}$$

is the power of the desired response,

$$\mathbf{d} \triangleq E\{\mathbf{x}y^*\} \tag{5.2.9}$$

is the cross-correlation vector between the data vector **x** and the desired response y, and

$$\mathbf{R} \triangleq E\{\mathbf{x}\mathbf{x}^H\} \tag{5.2.10}$$

is the correlation matrix of the data vector **x**. The matrix $\boldsymbol{R}$ is guaranteed to be Hermitian and nonnegative definite (see Section 2.2.4).

The function $P(\mathbf{c})$ is known as the *error performance surface* of the estimator. Equation (5.2.7) shows that the MSE $P(\mathbf{c})$ (1) depends *only* on the second-order moments of the desired response and the data and (2) is a quadratic function of the estimator coefficients and represents an $(M+1)$-dimensional surface with M degrees of freedom. We will see that if $\boldsymbol{R}$ is positive definite, then the quadratic function $P(\mathbf{c})$ is bowl-shaped and has a unique minimum that corresponds to the optimum parameters. The next example illustrates this fact for the second-order case.

EXAMPLE 5.2.1. If M=2 and the random variables $y, x_1,$ and x_2 are real-valued, the MSE is

$$P(c_1,c_2) = P_y - 2d_1c_1 - 2d_2c_2 + r_{11}c_1^2 + 2r_{12}c_1c_2 + r_{22}c_2^2$$

because $r_{12} = r_{21}$. And $P(c_1,c_2)$ is a second-order function of coefficients c_1 and c_2, and Figure 5.4 shows two plots of the function $P(c_1,c_2)$ that are quite different in appearance. The surface in Figure 5.4(*a*) looks like a bowl and has a unique extremum that is a minimum. The values for the error surface parameters are $P_y = 0.5,\ r_{11} = r_{22} = 4.5,\ r_{12} = r_{21} = -0.1545,\ d_1 = -0.5$.

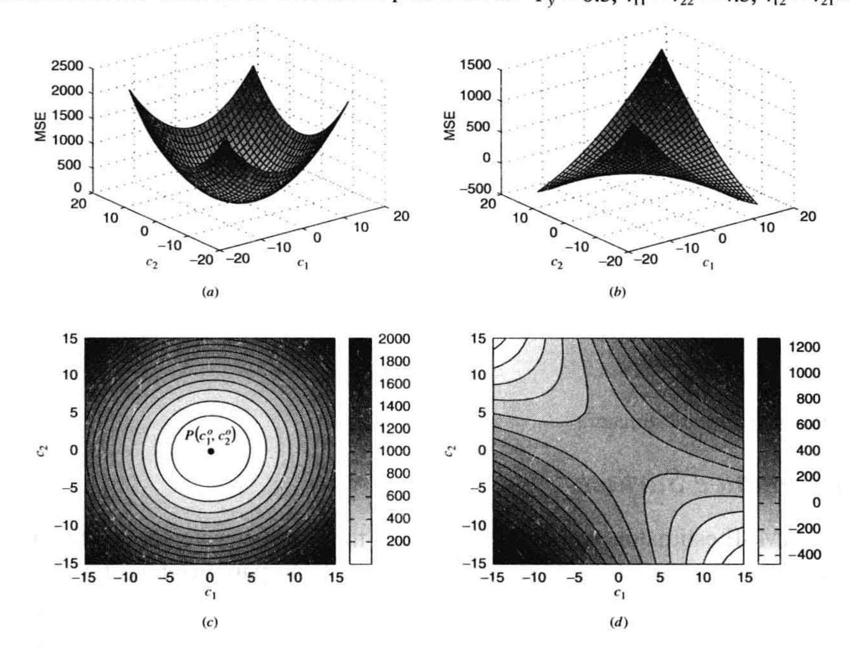


FIGURE5.4
Representative surface and contour plots for positive definite and negative definite quadratic error performance surfaces.

and $d_2 = -0.1545$. On the other hand, in Figure 5.4(*b*), we have a saddle point that is neither a minimum nor a maximum (here only the matrix elements have changed to $r_{11} = r_{22} = 1,\ r_{12} = r_{21} = 2$). If we cut the surfaces with planes parallel to the $(c_1,\ c_2)$ plane, we obtain *contours* of constant MSE that are shown in Figure 5.4(*c*) and (*d*). In conclusion, the error performance surface is bowl-shaped and has a unique minimum only if the matrix **R** is positive definite (the determinants of the two matrices are 20.23 and -3, respectively). Only in this case can we obtain an estimator that minimizes the MSE, and the contours are concentric ellipses whose center corresponds to the optimum estimator. The bottom of the bowl is determined by setting the partial derivatives with respect to the unknown parameters to zero, that is,

$$\frac{\partial P(c_1,c_2)}{\partial c_1} = 0 \quad \text{which results in} \quad r_{11}c_1^o + r_{12}c_2^o = d_1$$

$$\frac{\partial P(c_1,c_2)}{\partial c_2} = 0 \quad \text{which results in} \quad r_{12}c_1^o + r_{22}c_2^o = d_2$$

This is a linear system of two equations with two unknowns whose solution provides the coefficients c_1^o and c_2^o that minimize the MSE function $P(c_1, c_2)$.

When the optimum filter is specified by a rational system function, the error performance surface may be nonquadratic. This is illustrated in the following example.

EXAMPLE 5.2.2 Suppose that we wish to estimate the real-valued output $y(n)$ of the "unknown" system (see Figure 5.5)

$$G(z) = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}}$$

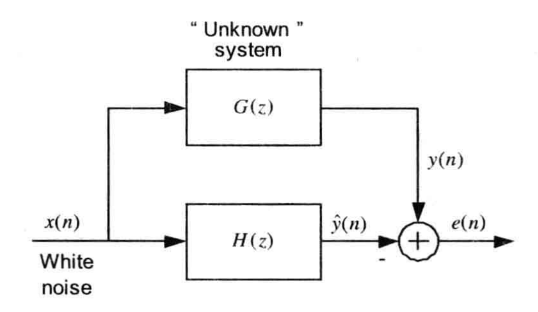


FIGURE 5.5
Identification of an "unknown" system using an optimum filter.

using the pole-zero filter

$$H(z) = \frac{b}{1 - az^{-1}}$$

by minimizing the MSE $E\{e^2(n)\}$ (Johnson and Larimore 1977). The input signal $x(n)$ is white noise with zero mean and variance σ_x^2. The MSE is given by

$$E\{e^2(n)\} = E\{[y(n) - \hat{y}(n)]^2\} = E\{y^2(n)\} - 2E\{y(n)\hat{y}(n)\} + E\{\hat{y}^2(n)\}$$

and is a function of parameters b and a. Since the impulse response $h(n) = ba^n u(n)$ of the optimum filter has infinite duration, we cannot use (5.2.7) to compute $E\{e^2(n)\}$ and to plot the error surface. The three components of $E\{e^2(n)\}$ can be evaluated as follows, using Parseval's theorem: The power of the desired response

$$E\{y^2(n)\} = \sigma_x^2 \sum_{n=0}^{\infty} g^2(n) = \frac{\sigma_x^2}{2\pi j} \oint G(z)G(z^{-1})z^{-1}\,dz \triangleq \sigma_x^2 \sigma_g^2$$

is constant and can be computed either numerically by using the first M "nonzero" samples of $g(n)$ or analytically by evaluating the integral using the residue theorem. The power of the optimum filter output is

$$E\{\hat{y}^2(n)\} = E\{x^2(n)\} \sum_{n=0}^{\infty} h^2(n) = \frac{\sigma_x^2}{2\pi j} \oint H(z)H(z^{-1})z^{-1}\,dz = \sigma_x^2 \frac{b^2}{1 - a^2}$$

which is a function of parameters b and a. The middle term is

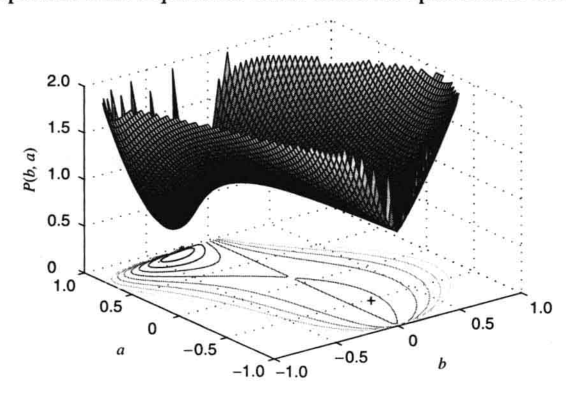
$$E\{y(n)\hat{y}(n)\} = E\left\{ \sum_{k=0}^{\infty} g(k)x(n-k) \sum_{m=0}^{\infty} h(m)x(n-m) \right\}$$

$$= \sigma_x^2 \sum_{k=0}^{\infty} g(k)h(k) = \frac{\sigma_x^2}{2\pi j} \oint_C G(z)H(z^{-1})z^{-1}\,dz = bG(z)|_{z^{-1}=a}$$

because $E\{x(n-k)x(n-m)\} = \sigma_x^2 \delta(m-k)$. For convenience we compute the normalized MSE

$$P(b,a) \triangleq \frac{E\{e^2(n)\}}{\sigma_g^2} = \sigma_x^2 - \frac{2b}{\sigma_g^2} G(z)\bigg|_{z^{-1}=a} + \frac{\sigma_x^2}{\sigma_g^2} \frac{b^2}{1 - a^2}$$

whose surface and contour plots are shown in Figure 5.6. We note that the resulting error performance surface is bimodal with a global minimum $P = 0.277$ at $(b, a) = (-0.311, 0.906)$ and a local minimum $P = 0.976$ at $(b, a) = (0.114, -0.519)$. As a result, the determination of the optimum filter requires the use of nonlinear optimization techniques with all associated drawbacks.



EIGURE 5.6
Illustration of the nonquadratic form of the error performance surface of a pole-zero optimum filter specified by the coefficients of its difference equation.

5.2.2 Derivation of the Linear MMSE Estimator

The approach in Example 5.2.1 can be generalized to obtain the necessary and sufficient conditions that determine the linear MMSE estimator.[1] Here, we present a simpler matrix-based approach that is sufficient for the scope of this chapter.

We first notice that we can put (5.2.7) into the form of a "perfect square" as

$$P(\boldsymbol{c}) = P_y - \boldsymbol{d}^{\mathrm{H}}\boldsymbol{R}^{-1}\boldsymbol{d} + (\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d})^{\mathrm{H}}\boldsymbol{R}^{-1}(\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d}) \tag{5.2.11}$$

where only the third term depends on $\boldsymbol{c}$. If $\boldsymbol{R}$ is positive definite, the inverse matrix $\boldsymbol{R}^{-1}$ exists and is positive definite; that is, $\boldsymbol{z}^{\mathrm{H}}\boldsymbol{R}^{-1}\boldsymbol{z} > 0$ for all $\boldsymbol{z} \neq 0$. Therefore, if $\boldsymbol{R}$ is positive definite, the term $\boldsymbol{d}^{\mathrm{H}}\boldsymbol{R}^{-1}\boldsymbol{d} > 0$ decreases the cost function by an amount determined exclusively by the second-order moments. In contrast, the term $(\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d})^{\mathrm{H}}\boldsymbol{R}^{-1}(\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d}) > 0$ increases the cost function depending on the choice of the estimator parameters. Thus, the best estimator is obtained by setting $\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d} = 0$.

Therefore, the *necessary and sufficient* conditions that determine the linear MMSE estimator $\mathbf{c}_o$ are

$$\boldsymbol{R}\boldsymbol{c}_o = \boldsymbol{d} \tag{5.2.12}$$

and $\boldsymbol{R}$ is positive definite (5.2.13)

In greater detail, (5.2.12) can be written as

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M1} & r_{M2} & \cdots & r_{MM} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix} \tag{5.2.14}$$

where

$$r_{ij} \triangleq E\{x_i x_j^*\} = r_{ji}^* \tag{5.2.15}$$

and

$$d_i \triangleq E\{x_i y^*\} \tag{5.2.16}$$

and are known as the set of *normal equations*. The invertibility of the correlation matrix $\boldsymbol{R}$ — and hence the existence of the optimum estimator — is guaranteed if $\boldsymbol{R}$ is positive definite. In theory, $\boldsymbol{R}$ is guaranteed to be nonnegative definite, but in physical applications it will almost always be positive definite. The normal equations can be solved by

[1] For complex-valued random variables, there are some complications that should be taken into account because $|e|^2$ is not an *analytic* function.

using any general purpose routine for a set of linear equations.

Using (5.2.11) and (5.2.12), we find that the MMSE P_o is

$$P_o = P_y - \mathbf{d}^H \mathbf{R}^{-1}\mathbf{d} = P_y - \mathbf{d}^H \mathbf{c}_o \tag{5.2.17}$$

where we can easily show that the term $\mathbf{d}^H \mathbf{c}_o$ is equal to $E\{|\hat{y}_o|^2\}$, the power of the optimum estimate. If $\mathbf{x}$ and y are uncorrelated $(\mathbf{d} = 0)$, we have the worst situation $(P_o = P_y)$ because there is no linear estimator that can reduce the MSE. If $\mathbf{d} \neq 0$, there is always going to be some reduction in the MSE owing to the correlation between the data vector $\mathbf{x}$ and the desired response y, assuming that $\mathbf{R}$ is positive definite. The best situation corresponds to $\hat{y} = y$, which gives $P_o = 0$. Thus, for comparison purposes, we use the *normalized MSE*

$$\varepsilon \triangleq \frac{P_o}{P_y} = 1 - \frac{P_{\hat{y}_o}}{P_y} \tag{5.2.18}$$

because it is bounded between 0 and 1, that is,

$$0 \le \varepsilon \le 1 \tag{5.2.19}$$

If $\tilde{\mathbf{c}}$ is the deviation from the optimum vector $\mathbf{c}_o$, that is, if $\mathbf{c} = \mathbf{c}_o + \tilde{\mathbf{c}}$, then substituting into (5.2.11) and using (5.2.17), we obtain

$$P(\mathbf{c}_o + \tilde{\mathbf{c}}) = P(\mathbf{c}_o) + \tilde{\mathbf{c}}^H \mathbf{R}\tilde{\mathbf{c}} \tag{5.2.20}$$

Equation (5.2.20) shows that if $\mathbf{R}$ is positive definite, any deviation $\tilde{\mathbf{c}}$ from the optimum vector $\mathbf{c}_o$ increases the MSE by an amount $\tilde{\mathbf{c}}^H \mathbf{R}\tilde{\mathbf{c}} > 0$, which is known as the *excess MSE*, that is,

$$\text{Excess MSE} \triangleq P(\mathbf{c}_o + \tilde{\mathbf{c}}) - P(\mathbf{c}_o) = \tilde{\mathbf{c}}^H \mathbf{R}\tilde{\mathbf{c}} \tag{5.2.21}$$

We emphasize that the excess MSE depends only on the input correlation matrix and not on the desired response. This fact has important implications because any deviation from the optimum can be detected by monitoring the MSE.

For nonzero-mean random variables, we use the estimator $\hat{y} \triangleq c_0 + \mathbf{c}^H \mathbf{x}$. The elements of $\mathbf{R}$ and $\mathbf{d}$ are replaced by the corresponding covariances and $c_0 = E\{y\} - \mathbf{c}^H E\{\mathbf{x}\}$ (see Problem 5.1). In the sequel, unless otherwise explicitly stated, we assume that all random variables have zero mean or have been reduced to zero mean by replacing y by $y - E\{y\}$ and $\mathbf{x}$ by $\mathbf{x} - E\{\mathbf{x}\}$.

5.2.3 Principal-Component Analysis of the Optimum Linear Estimator

The properties of optimum linear estimators and their error performance surfaces depend on the correlation matrix $\mathbf{R}$. We can learn a lot about the nature of the optimum estimator if we express $\mathbf{R}$ in terms of its eigenvalues and eigenvectors. Indeed, from Section 2.2.4, we have

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^H = \sum_{i=1}^{M} \lambda_i \mathbf{q}_i \mathbf{q}_i^H \qquad \text{and} \qquad \Lambda = \mathbf{Q}^H \mathbf{R}\mathbf{Q} \tag{5.2.22}$$

where

$$\Lambda = \text{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_M\} \tag{5.2.23}$$

are the eigenvaluse of $\mathbf{R}$, assumed to be distinct, and

$$\mathbf{Q} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_M] \tag{5.2.24}$$

are the eigenvectors of $\mathbf{R}$. The modal matrix $\mathbf{Q}$ is unitary, that is,

$$\mathbf{Q}^H \mathbf{Q} = \mathbf{I} \tag{5.2.25}$$

which implies that $\mathbf{Q}^{-1} = \mathbf{Q}^H$. The relationship (5.2.22) between $\mathbf{R}$ and Λ is known as a *similarity transformation.*

In general, the multiplication of a vector by a matrix changes both the length and the direction of the vector. We define a coordinate transformation of the optimum parameter vector by

$$\mathbf{c}_o' \triangleq \mathbf{Q}^H \mathbf{c}_o \qquad \text{or} \qquad \mathbf{c}_o \triangleq \mathbf{Q}\mathbf{c}_o' \tag{5.2.26}$$

Since
$$\|c_o\| = (Qc'_o)^H Qc'_o = c'^H_o Q^H Qc'_o = \|c'_o\| \tag{5.2.27}$$

the transformation (5.2.26) changes the direction of the transformed vector but not its length.

If we substitute (5.2.22) into the normal equations (5.2.12), we obtain

$$Q\Lambda Q^H c_o = d \qquad \text{or} \qquad \Lambda Q^H c_o = Q^H d$$

which results in

$$\Lambda c'_o = d' \tag{5.2.28}$$

where
$$d' \triangleq Q^H d \qquad \text{or} \qquad d \triangleq Qd' \tag{5.2.29}$$

is the transformed "decoupled" cross-correlation vector.

Because Λ is diagonal, the set of M equations (5.2.8) can be written as

$$\lambda_i c'_{o,i} = d'_i \qquad 1 \le i \le M \tag{5.2.30}$$

where $c'_{o,i}$ and d'_i are the components of $\mathbf{c}'_o$ and $\mathbf{d}'$, respectively. This is an uncoupled set of M first-order equations. If $\lambda_i \ne 0$, then

$$c'_{o,i} = \frac{d'_i}{\lambda_i} \qquad 1 \le i \le M \tag{5.2.31}$$

and if $\lambda_i = 0$, the value of $c'_{o,i}$ is indeterminate.

The MMSE becomes

$$\begin{aligned} P_o &= P_y - d^H c_o \\ &= P_y - (Qd')^H Qc'_o = P_y - d'^H c'_o \\ &= P_y - \sum_{i=1}^{M} d'^*_i c'_{o,i} = P_y - \sum_{i=1}^{M} \frac{|d'_i|^2}{\lambda_i} \end{aligned} \tag{5.2.32}$$

which shows how the eigenvalues and the decoupled cross-correlations affect the performance of the optimum filter. The advantage of (5.2.31) and (5.2.32) is that we can study the behavior of each parameter of the optimum estimator independently of all the remaining ones.

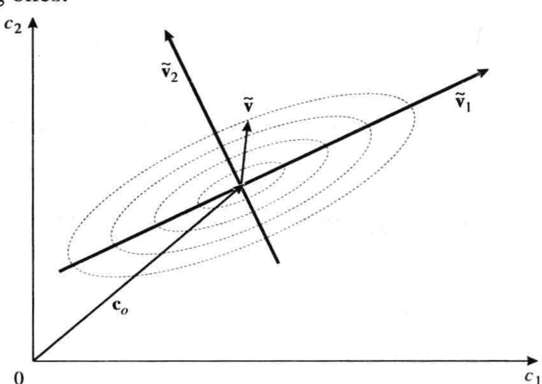


FIGURE 5.7
Contours of constant MSE and principal-component axes for a second-order quadratic error surface.

To appreciate the significance of the principal-component transformation, we will discuss the error surface of a second-order estimator. However, all the results can be easily generalized to estimators of order M, whose error performance surface exists in a space of M+1 dimensions. Figure 5.7 shows the contours of constant MSE for a positive definite, second-order error surface. The contours are concentric ellipses centered at the tip of the optimum vector $\mathbf{c}_o$. We define a new coordinate system with origin at $\mathbf{c}_o$ and axes determined by the major axis $\tilde{\mathbf{v}}_1$ and the minor axis $\tilde{\mathbf{v}}_2$ of the ellipses. The two axes are orthogonal, and the resulting system is known as the *principal*

coordinate system. The transformation from the "old" system to the "new" system is done in two steps:

$$\begin{aligned} \text{Translation :} \quad & \tilde{c} = c - c_o \\ \text{Rotation :} \quad & \tilde{v} = Q^{\mathrm{H}}\tilde{c} \end{aligned} \tag{5.2.33}$$

where the rotation changes the axes of the space to match the axes of the ellipsoid. The excess MSE (5.2.21) becomes

$$\Delta P(\tilde{\mathbf{v}}) = \tilde{c}^{\mathrm{H}}\mathbf{R}\tilde{\mathbf{c}} = \tilde{\mathbf{c}}^{\mathrm{H}}\mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{H}}\tilde{\mathbf{c}} = \tilde{\mathbf{v}}^{\mathrm{H}}\Lambda\tilde{v} = \sum_{i=1}^{M}\lambda_i\,|\,\tilde{v}_i\,|^2 \tag{5.2.34}$$

which shows that the penalty paid for the deviation of a parameter from its optimum value is proportional to the corresponding eigenvalue. Clearly, changes in uncoupled parameters (which correspond to $\lambda_i = 0$) do not affect the excess MSE.

Using (5.2.22), we have

$$c_o = R^{-1}d = Q\Lambda^{-1}Q^H d = \sum_{i=1}^{M}\frac{q_i^{\mathrm{H}}d}{\lambda_i}q_i = \sum_{i=1}^{M}\frac{d_i'}{\lambda_i}q_i \tag{5.2.35}$$

and the optimum estimate can be written as

$$\hat{y}_{\mathrm{o}} = \mathbf{c}_{\mathrm{o}}^{\mathrm{H}}\mathbf{x} = \sum_{i=1}^{M}\frac{d_i'}{\lambda_i}(q_i^H\,\mathbf{x}) \tag{5.2.36}$$

which leads to the representation of the optimum estimator shown in Figure 5.8. The eigenfilters q_i decorrelate the data vector $\mathbf{x}$ into its principal components, which are weighted and added to produce the optimum estimate.

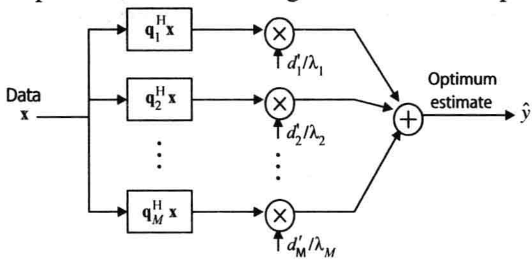


FIGURE 5.8
Principal-components representation of the optimum linear estimator.

5.2.4 *Geometric Interpretations and the Principle of Orthogonality*

It is convenient and pedagogic to think of random variables with zero mean value and finite variance as vectors in an abstract vector space with an inner product (i.e., a Hilbert space) defined by their correlation

$$< x, y > \triangleq E\{xy^*\} \tag{5.2.37}$$

and the length of a vector by

$$\|x\|^2 \triangleq < x, x > = E\{|\,x\,|^2\} < \infty \tag{5.2.38}$$

From the definition of the correlation coefficient and the above definitions, we obtain

$$|< x, y >|^2 \;\le\; \|x\|\|y\| \tag{5.2.39}$$

which is known as the Cauchy-Schwartz inequality. Two random variables are *orthogonal*, denoted by $x \perp y$, if

$$< x, y > = E\{xy^*\} = 0 \tag{5.2.40}$$

which implies they are uncorrelated since they have zero mean.

This geometric viewpoint offers an illuminating and intuitive interpretation for many aspects of MSE estimation that we will find very useful. Indeed, using (5.2.9), (5.2.10), and (5.2.12), we have

$$E\{\boldsymbol{x}e_o^*\} = E\{\boldsymbol{x}(y^* - \boldsymbol{x}^H\boldsymbol{c}_o)\} = E\{\boldsymbol{x}y^*\} - E\{\boldsymbol{x}\boldsymbol{x}^H\}\boldsymbol{c}_o = \boldsymbol{d} - \boldsymbol{R}\boldsymbol{c}_o = 0$$

Therefore

$$E\{\boldsymbol{x}e_o^*\} = \boldsymbol{0} \tag{5.2.41}$$

or

$$E\{x_m e_o^*\} = 0 \qquad \text{for } 1 \le m \le M \tag{5.2.42}$$

that is, the *estimation error is orthogonal to the data used for the estimation*. Equations (5.2.41), or equivalently (5.2.42), are known as the *orthogonality principle* and are widely used in linear MMSE estimation.

To illustrate the use of the orthogonality principle, we note that any linear combination $c_1^* x_1 + \cdots + c_M^* x_M$ lies in the subspace defined by the vectors[2] $x_1, \ldots, x_M$. Therefore, the estimate $\hat{y}$ that minimizes the squared length of the error vector e, that is, the MSE, is determined by the foot of the perpendicular from the tip of the vector y to the "plane" defined by vectors $x_1, \ldots, x_M$. This is illustrated in Figure 5.9 for $M = 2$. Since e_o is perpendicular to every vector in the plane, we have $x_m \perp e_o$, $1 \le m \le M$, which leads to the orthogonality principle (5.2.42). Conversely, we can start with the orthogonality principle (5.2.4) and derive the normal equations. This interpretation has led to the name normal equations for (5.2.12). We will see several times that the concept of orthogonality has many important theoretical and practical implications. As an illustration, we apply the Pythagorean theorem to the orthogonal triangle formed by vectors $\hat{y}_o$, e_o, and y, in Figure 5.9, to obtain

$$\|y\|^2 = \|\hat{y}_o\|^2 + \|e_o\|^2$$

or

$$E\{|y|^2\} = E\{|\hat{y}_o|^2\} + E\{|e_o|^2\} \tag{5.2.43}$$

which decomposes the power of the desired response into two component, one that is correlated to the data and one that is uncorrelated to thd data.

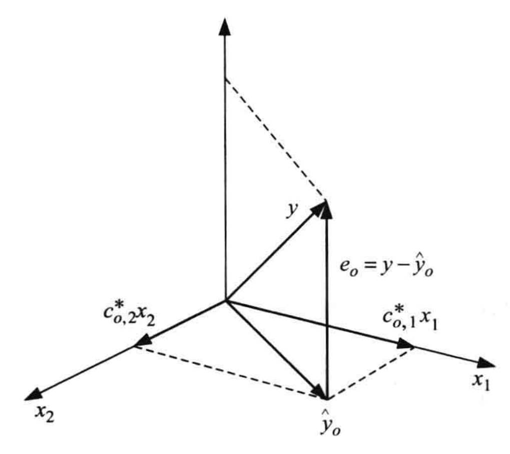


FIGURE 5.9
Pictorial illustration of the orthogonality principle

5.2.5 *Summary and Further Properties*

We next summarize, for emphasis and future reference, some important properties of optimum, in the MMSE sense, linear estimators.

1. Equations (5.2.12) and (5.2.17) show that the optimum estimator and the MMSE depend *only* on the second-order moments of the desired response and the data. The dependence on the second-order moments is a consequence of both the linearity of the estimator and the use of the MSE criterion.
2. The error performance surface of the optimum estimator is a quadratic function of its coefficients. If the data correlation matrix is positive definite, this function has a unique minimum that determines the optimum set of coefficients. The surface can be visualized as a bowl, and the optimum estimator corresponds to the bottom of the

[2] We should be careful to avoid confusing vector random variables, that is, vectors whose components are random variables, and random variables interpreted as vectors in the abstract vector space defined by Equations (5.2.37) to (5.2.39).

bowl.

3. If the data correlation matrix $\boldsymbol{R}$ is positive definite, any deviation from the optimum increases the MMSE according to (5.2.21). The resulting excess MSE depends on $\boldsymbol{R}$ only. This property is very useful in the design of adaptive filters.
4. When the estimator operates with the optimum set of coefficients, the error e_o is uncorrelated (orthogonal) to both the data $x_1, x_2, \ldots, x_M$ and the optimum estimate $\hat{y}_o$. This property is very useful if we want to monitor the performance of an optimum estimator in practice and is used also to design adaptive filters.
5. The MMSE, the optimum estimator, and the optimum estimate can be expressed in terms of the eigenvalues and eigenvectors of the data correlation matrix. See (5.2.32), (5.2.35), and (5.2.36).
6. The general (unconstrained) estimator

$$\hat{y} \triangleq h(\mathbf{x}) = h(x_1, x_2, \cdots, x_M)$$

that minimizes the MSE

$$P = E\{|y - h(\mathbf{x})|^2\}$$

with respect to $h(\mathbf{x})$ is given by the *mean of the conditional density*, that is,

$$\hat{y}_o \triangleq h_o(\mathbf{x}) = E\{y \mid \mathbf{x}\} = \int_{-\infty}^{\infty} y p_y(y \mid \mathbf{x})\, dy$$

and clearly is a nonlinear function of $x_1, \ldots, x_M$. If the desired response and the data are jointly Gaussian, the linear MMSE estimator is the best in the MMSE sense; that is, we *cannot* find a nonlinear estimator that produces an estimate with smaller MMSE (Papoulis 1991).

5.3 Optimum Finite Impulse Response Filters

In the previous section, we presented the theory of general linear MMSE estimators [see Figure 5.1(*a*)]. In this section, we apply these results to the design of optimum linear filters, that is, filters whose performance is the best possible when measured according to the MMSE criterion [see Figure 5.1(*b*)]. The general formulation of the optimum filtering problem is shown in Figure 5.10. The optimum filter forms an estimate $\hat{y}(n)$ of the desired response $y(n)$ by using samples from a related input signal $x(n)$. The theory of optimum filters was developed by Wiener (1942) in continuous time and Kolmogorov (1939) in discrete time. Levinson (1947) reformulated the theory for FIR filters and stationary processes and developed an efficient algorithm for the solution of the normal equations that exploits the Toeplitz structure of the autocorrelation matrix $\boldsymbol{R}$ (see Section 6.4). For this reason, linear MMSE filters are often referred to as Wiener filters.

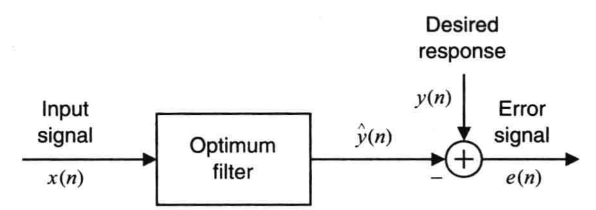


FIGURE 5.10
Block diagram representation of the optimum filtering problem.

We consider a linear FIR filter specified by its impulse response $h(n,k)$. The output of the filter is determined by the superposition summation

$$\hat{y}(n) = \sum_{k=0}^{M-1} h(n,k)x(n-k) \tag{5.3.1}$$

$$\triangleq \sum_{k=1}^{M} c_k^*(n)x(n-k+1) \triangleq \mathbf{c}^H(n)\mathbf{x}(n) \tag{5.3.2}$$

where

$$\boldsymbol{c(n)} \triangleq [c_1(n)\ c_2(n)\ \cdots\ c_M(n)]^{\mathrm{T}} \tag{5.3.3}$$

and

$$\boldsymbol{x(n)} \triangleq [x(n)\ x(n-1)\ \cdots\ x(n-M+1)]^{\mathrm{T}} \tag{5.3.4}$$

are the filter *coefficient vector*[3] and the *input data vector*, respectively. Equation (5.3.1) becomes a convolution if $h(n,k)$ does not depend on n, that is, when the filter is time-invariant. The objective is to find the coefficient vector that minimizes the MSE $E\{|e(n)|^2\}$.

We prefer FIR over IIR filters because (1) any stable IIR filter can be approximated to any desirable degree by an FIR filter and (2) optimum FIR filters are easily obtained by solving a linear system of equations.

5.3.1 Design and Properties

To determine the optimum FIR filter $\mathbf{c}_o(n)$, we note that at every time instant *n*, the optimum filter is the linear MMSE estimator of the desired response $y(n)$ based on the data $\boldsymbol{x}(n)$. Since for any fixed *n* the quantities $y(n), x(n), \ldots, x(n-M+1)$ are random variables, we can determine the optimum filter either from (5.2.12) by replacing $\boldsymbol{x}$ by $\mathbf{x}(n)$, y by $y(n)$, and $\boldsymbol{c}_o$ by $\boldsymbol{c}_o(n)$; or by applying the orthogonality principle (5.2.41). Indeed, using (5.3.41), (5.1.2), and (5.3.2), we have

$$E\{\boldsymbol{x}(n)[y^*(n)-\boldsymbol{x}^{\mathrm{H}}(n)\boldsymbol{c}_o(n)]\}=0 \tag{5.3.5}$$

which leads to the following set of normal equations

$$\boldsymbol{R}(n)\boldsymbol{c}_o(n)=\boldsymbol{d}(n) \tag{5.3.6}$$

where

$$\boldsymbol{R}(n)\triangleq E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\} \tag{5.3.7}$$

is the correlation matrix of the input data vector and

$$\boldsymbol{d}(n)\triangleq E\{\boldsymbol{x}(n)y^*(n)\} \tag{5.3.8}$$

is the cross-correlation vector between the desired response and the input data vector, that is, the input values stored currently in the filter memory and used by the filter to estimate the desired response. We see that, at every time *n*, the coefficients of the optimum filter are obtained as the solution of a linear system of equations. The filter $\boldsymbol{c}_o(n)$ is optimum if and only if the Hermitian matrix $\boldsymbol{R}(n)$ is positive definite.

To find the MMSE, we can use either (5.2.17) or the orthogonality principle (5.2.41). Using the orthogonality principle, we have

$$\begin{aligned} P_o(n) &= E\{e_o(n)[y^*(n)-\boldsymbol{x}^{\mathrm{H}}(n)\boldsymbol{c}_o(n)]\} \\ &= E\{e_o(n)y^*(n)\} \qquad \text{due to orthogonality} \\ &= E\{[y(n)-\boldsymbol{x}^H(n)\boldsymbol{c}_o(n)]y^*(n)\} \end{aligned}$$

which can be written as

$$P_o(n)=P_y(n)-\boldsymbol{d}^{\mathrm{H}}(n)\boldsymbol{c}_o(n) \tag{5.3.9}$$

The first term

$$P_y(n)\triangleq E\{|y(n)|^2\} \tag{5.3.10}$$

is the power of the desired response signal and represents the MSE in the absence of filtering. The second term $\boldsymbol{d}^H(n)\boldsymbol{c}_o(n)$ is the reduction in the MSE that is obtained by using the optimum filter.

In many practical applications, we need to know the performance of the optimum filter in terms of MSE reduction prior to computing the coefficients of the filter. Then we can decide if it is preferable to (1) use an optimum filter (assuming we can design one), (2) use a simpler suboptimum filter with adequate performance, or (3) not use a filter at all. Hence, the performance of the optimum filter can serve as a yardstick for other competing methods.

The optimum filter consists of (1) a linear system solver that determines the optimum set of coefficients from the normal equations formed, using the known second-order moments, and (2) a discrete-time filter that computes the estimate $\hat{y}(n)$ (see Figure 5.11). The solution of (5.3.6) can be obtained by using standard linear system solution techniques. In MATLAB, we solve (5.3.6) by `copt=R\d` and compute the MMSE by `Popt=Py-dot(conj(d),copt)`. The

[3] We define $c_k(n)\triangleq h^*(n,k)$ in order to comply with the definition $\boldsymbol{R}(n)\triangleq E\{\boldsymbol{x}(n)\boldsymbol{x}^H(n)\}$ of the correlation matrix.

optimum filter is implemented by `yest=filter(copt,1,x)`. We emphasize that the optimum filter only needs the input signal for its operation, that is, to form the estimate of $y(n)$; the desired response, if it is available, may be used for other purposes.

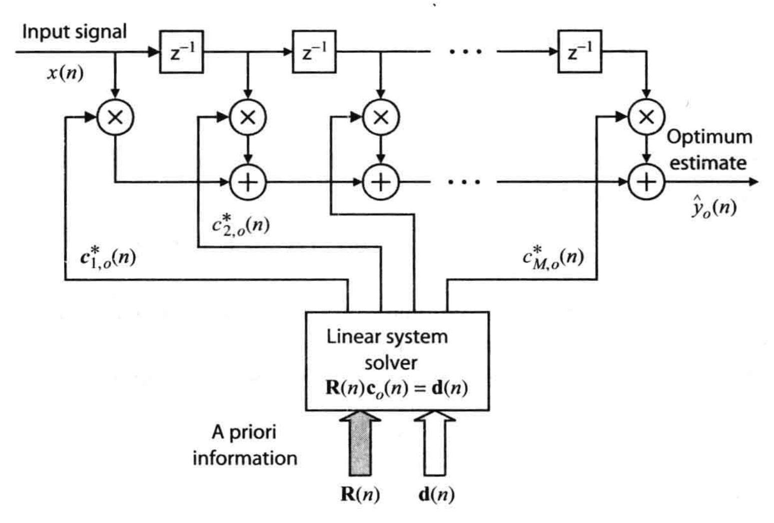


FIGURE 5.11
Design and implementation of a time-varying optimum FIR filter.

Conventional frequency-selective filters are designed to shape the spectrum of the input signal within a specific frequency band in which it operates. In this sense, these filters are effective only if the components of interest in the input signal have their energy concentrated within *nonoverlapping* bands. To design the filters, we need to know the limits of these bands, not the values of the sequences to be filtered. Note that such filters do not depend on the values of the data (values of the samples) to be filtered; that is, they are *not* data-adaptive. In contrast, optimum filters are designed using the second-order moments of the processed signals and have the same effect on all classes of signals with the same second-order moments. Optimum filters are effective even if the signals of interest have overlapping spectra. Although the actual data values also do not affect optimum filters, that is, they are also not data-adaptive, these filters are optimized to the statistics of the data and thus provide superior performance when judged by the statistical criterion.

The dependence of the optimum filter *only* on the second-order moments is a consequence of the linearity of the filter and the use of the MSE criterion. Phase information about the input signal or non-second-order moments of the input and desired response processes is not needed; even if the moments are known, they are not used by the filter. Such information is useful only if we employ a nonlinear filter or use another criterion of performance.

The error performance surface of the optimum direct-form FIR filter is a quadratic function of its impulse response. If the input correlation matrix is positive definite, this function has a unique minimum that determines the optimum set of coefficients. The surface can be visualized as a bowl, and the optimum filter corresponds to the bottom of the bowl. The bottom is moving if the processes are nonstationary and fixed if they are stationary. In general, the shape of the error performance surface depends on the criterion of performance and the structure of the filter. Note that the use of another criterion of performance or another filter structure may lead to error performance surfaces with multiple local minima or saddle points.

5.3.2 Optimum FIR Filters for Stationary Processes

Further simplifications and additional insight into the operation of optimum linear filters are possible when the input and desired response stochastic processes are *jointly wide-sense stationary*. In this case, the correlation matrix of the input data and the cross-correlation vector do *not* depend on the time index *n*. Therefore, the optimum filter and the MMSE are *time-invariant* (i.e., they are independent of the time index n) and are determined by

$$\boldsymbol{R}\boldsymbol{c}_o = \boldsymbol{d} \tag{5.3.11}$$

and

$$P_o = P_y - \boldsymbol{d}^{\mathrm{H}}\boldsymbol{c}_o \tag{5.3.12}$$

Owing to stationarity, the autocorrelation matrix is

$$\boldsymbol{R} \triangleq \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(M-1) \\ r_x^*(1) & r_x(0) & \cdots & r_x(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_x^*(M-1) & r_x^*(M-2) & \cdots & r_x(0) \end{bmatrix} \tag{5.3.13}$$

determined by the autocorrelation $r_x(l) = E\{x(n)x^*(n-l)\}$ of the input signal. The cross-correlation vector between the desired response and the input data vector is

$$\boldsymbol{d} \triangleq [d_1\ d_2\ \cdots\ d_M]^{\mathrm{T}} \triangleq [r_{yx}(0)\ r_{yx}(1)\ \cdots\ r_{yx}(M-1)]^{\mathrm{T}} \tag{5.3.14}$$

and P_y is the power of the desired response. For stationary processes, the matrix **R** is Toeplitz and positive definite unless the components of the data vector are linearly dependent.

Since the optimum filter is time-invariant, it is implemented by using convolution

$$\hat{y}_o(n) = \sum_{k=0}^{M-1} h_o(k)x(n-k) \tag{5.3.15}$$

where $h_o(n) = c_{o,n}^*$ is the impulse response of the optimum filter.

Using (5.3.13), (5.3.14), $h_o(n) = c_{o,n}^*$, and $r(l) = r^*(-l)$, we can write the normal equations (5.3.11) more explicitly as

$$\sum_{k=0}^{M-1} h_o(k)r(m-k) = r_{yx}(m) \qquad 0 \le m \le M-1 \tag{5.3.16}$$

which is the discrete-time counterpart of the *Wiener-Hopf* integral equation, and its solution determines the impulse response of the optimum filter. We notice that the cross-correlation between the input signal and the desired response (right-hand side) is equal to the convolution between the autocorrelation of the input signal and the optimum filter (left-hand side). Thus, to obtain the optimum filter, we need to solve a convolution equation.

The MMSE is given by

$$P_o = P_y - \sum_{k=0}^{M-1} h_o(k)r_{yx}^*(k) \tag{5.3.17}$$

which is obtained by substituting (5.3.14) into (5.3.12). Table 5.2 summarizes the information required for the design of an optimum (in the MMSE sense) linear time-invariant filter, the Wiener-Hopf equations that define the filter, and the resulting MMSE.

TABLE 5.2.
Specification of optimum linear filters for stationary signals. The limits 0 and $M-1$ on the summations can be replaced by any values M_1 and M_2.

Filter and Error Definitions	$e(n) \triangleq y(n) - \sum_{k=0}^{M-1} h(k)x(n-k)$
Criterion of Performance	$P \triangleq E\{\lvert e(n)\rvert^2\} \to$ minimum
Wiener-Hopf Equations	$\sum_{k=0}^{M-1} h_o(k)r_x(m-k) = r_{yx}(m)$, $0 \le m \le M-1$
Minimum MSE	$P_o = P_y - \sum_{k=0}^{M-1} h_o(k)r_{yx}^*(k)$
Second-Order Statistics	$r_x(l) = E\{x(n)x^*(n-l)\}, P_y = \{ly(n)l^2\}$ $r_{yx}(l) = E\{y(n)x^*(n-l)\}$

To summarize, for nonstationary processes $\mathbf{R}(n)$ is Hermitian and nonnegative definite, and the optimum filter $\mathbf{h}_o(n)$ is time-varying. For stationary processes, $\boldsymbol{R}$ is Hermitian, nonnegative definite, and Toeplitz, and the optimum filter is time-invariant. A Toeplitz autocorrelation matrix is positive definite if the power spectrum of the input satisfies $R_x(e^{j\omega}) > 0$ for all frequencies ω. In both cases, the filter is used for all realizations of the processes. If $M = \infty$, we have a causal IIR optimum filter determined by an infinite-order

linear system of equations that can only be solved in the stationary case by using analytical techniques (see Section 5.6).

EXAMPLE 5.3.1 Consider a harmonic random process

$$y(n) = A\cos(\omega_0 n + \phi)$$

with fixed, but unknown, amplitude and frequency, and random phase ϕ, uniformly distributed on the interval from 0 to 2π. This process is corrupted by additive white Gaussian noise $v(n) \sim N(0, \sigma_v^2)$ that is uncorrelated with $y(n)$. The resulting signal $x(n) = y(n) + v(n)$ is available to the user for processing. Design an optimum FIR filter to remove the corrupting noise $v(n)$ from the observed signal $x(n)$.

Solution. The input of the optimum filter is $x(n)$, and the desired response is $y(n)$. The signal $y(n)$ is obviously unavailable, but to design the filter, we only need the second-order moments $r_x(l)$ and $r_{yx}(l)$. We first note that since $y(n)$ and $v(n)$ are uncorrelated, the autocorrelation of the input signal is

$$r_x(l) = r_y(l) + r_v(l) = \frac{1}{2}A^2 \cos\omega_0 l + \sigma_v^2 \delta(l)$$

where $r_y(l) = 1/2A^2 \cos\omega_0 l$ is the autocorrelation of $y(n)$. The cross-correlation between the desired response $y(n)$ and the input signal $x(n)$ is

$$r_{yx}(l) = E\{y(n)[y(n-l) + v(n-l)]\} = r_y(l)$$

Therefore, the autocorrelation matrix $\boldsymbol{R}$ is symmetric Toeplitz and is determined by the elements $r(0), r(1), \ldots, r(M-1)$ of its first row. The right-hand side of the Wiener-Hopf equations is $\mathbf{d} = [r_y(0)\ r_y(1)\ \cdots\ r_y(M-1)]^{\mathrm{T}}$. If we know $r_y(l)$ and σ_v^2, we can numerically determine the optimum filter and the MMSE from (5.3.11) and (5.3.12). For example, suppose that $A = 0.5$, $f_0 = \omega_0/(2\pi) = 0.05$, and $\sigma_v^2 = 0.5$. The input signal-to-noise ratio (SNR) is

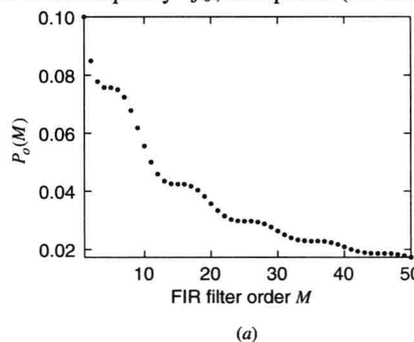
$$\text{SNR}_I = 10\log\frac{A^2/2}{\sigma_v^2} = -6.02 \text{ dB}$$

The *processing gain* (*PG*), defined as the ratio of signal-to-noise ratios at the output and input of a signal processing system

$$\text{PG} \triangleq \frac{\text{SNR}_O}{\text{SNR}_I}$$

provides another useful measure of performance.

The first problem we encounter is how to choose the order M of the filter. In the absence of any a priori information, we compute $\boldsymbol{h}_o$ and P_o^h for $1 \le M \le M_{\max} = 50$ and PG and plot both results in Figure 5.1.2. We see that an $M = 20$ order filter provides satisfactory performance. Figure 5.13 shows a realization of the corrupted and filtered signals. Another useful approach to evaluate how well the optimum filter enhances a harmonic signal is to compute the spectra of the input and output signals and the frequency response of the optimum filter. These are shown in Figure 5.14, where we see that the optimum filter has a sharp bandpass about frequency f_0, as expected (for details see Problem 5.5).



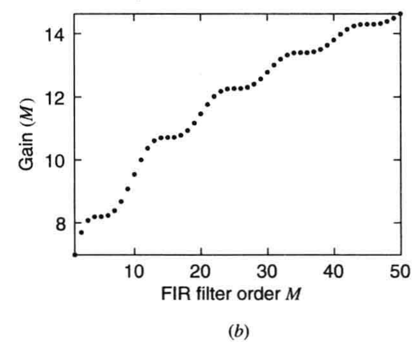


FIGURE 5.12
Plots of (*a*) the MMSE and (*b*) the processing gain as a function of the filter order M.

To illustrate the meaning of the estimator's optimality, we will use a Monte Carlo simulation. Thus, we generate $K = 100$ realizations of the sequence $x(\zeta_i, n), 0 \le n \le N-1$ $(N = 1000)$; we compute the output sequence $\hat{y}(\zeta_i, n)$, using (5.3.15); and then the error sequence $e(\zeta_i, n) = y(\zeta_i, n) - \hat{y}(\zeta_i, n)$ and its variance $\hat{P}(\zeta_i)$. Figure (5.15) shows a plot of. $\hat{P}(\zeta_i)$, $1 \le \zeta_i \le K$. We notice that although the filter performs better or worse than the optimum in particular cases, on average its performance is close to the theoretically predicted one. This is exactly the meaning of the MMSE criterion: *optimum*

performance on the average (*in the MMSE sense*) .

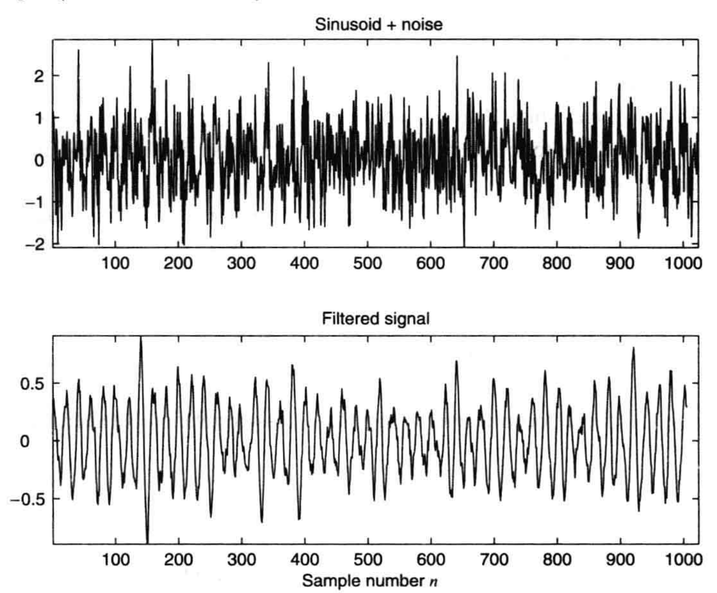


FIGURE 5.13
Example of the noise-corrupted and filtered sinusoidal signals.

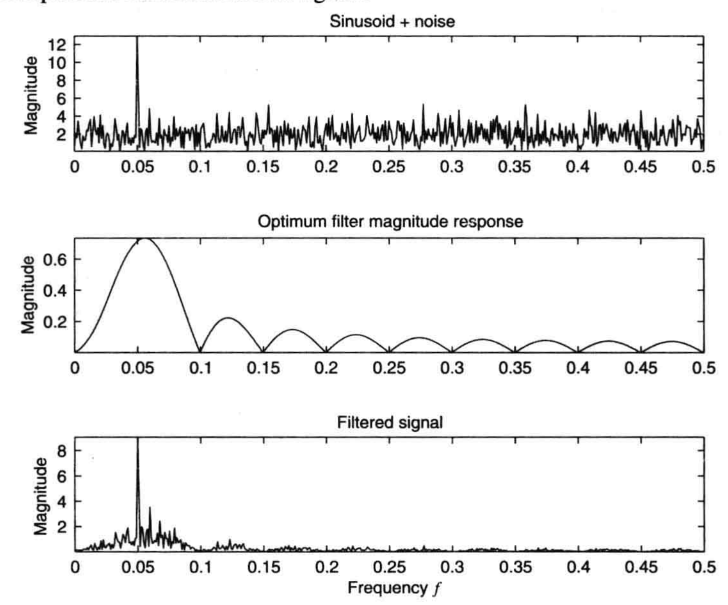


FIGURE 5.14
PSD of the input signal, magnitude response of the optimum filter, and PSD of the output signal.

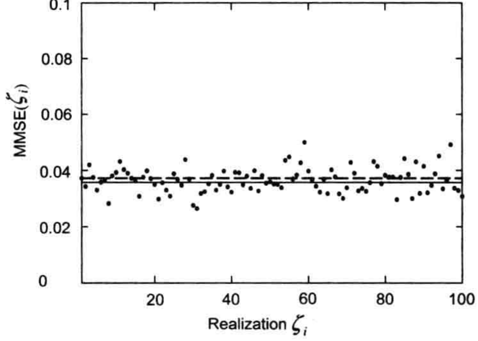


FIGURE 5.15
Results of Monte Carlo simulation of the optimum filter.

For a certain realization, the optimum filter may not perform as well as some other linear filters; however, on average, it performs better than any other linear filter of the same order when all possible realizations of $x(n)$ and $y(n)$ are considered.

5.3.3 *Frequency-Domain Interpretations*

We will now investigate the performance of the optimum filter, for stationary processes, in the frequency domain. Using (5.2.7), (5.3.13), and (5.3.14), we can easily show that the MSE of an FIR filter $h(n)$ is given by

$$P = E\{|e(n)|^2\} = r_y(0) - \sum_{k=0}^{M-1} h(k)r_{yx}^*(k) - \sum_{k=0}^{M-1} h^*(k)r_{yx}(k) + \sum_{k=0}^{M-1}\sum_{l=0}^{M-1} h(k)r(l-k)h^*(l) \tag{5.3.18}$$

The frequency response function of the FIR filter is

$$H(\mathrm{e}^{\mathrm{j}\omega}) \triangleq \sum_{k=0}^{M-1} h(k)\mathrm{e}^{-\mathrm{j}\omega k} \tag{5.3.19}$$

Using Parseval's theorem,

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X_1(\mathrm{e}^{\mathrm{j}\omega})X_2^*(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega \tag{5.3.20}$$

we can show that the MSE (6.4.18) can be expressed in the frequency domain as

$$P = r_y(0) - \frac{1}{2\pi}\int_{-\pi}^{\pi} [H(\mathrm{e}^{\mathrm{j}\omega})R_{yx}^*(\mathrm{e}^{\mathrm{j}\omega}) + H^*(\mathrm{e}^{\mathrm{j}\omega})R_{yx}(\mathrm{e}^{\mathrm{j}\omega}) - H(\mathrm{e}^{\mathrm{j}\omega})H^*(\mathrm{e}^{\mathrm{j}\omega})R_x(\mathrm{e}^{\mathrm{j}\omega})]\,d\omega \tag{5.3.21}$$

where $R_x(\mathrm{e}^{\mathrm{j}\omega})$ is the PSD of $x(n)$ and $R_{yx}(\mathrm{e}^{\mathrm{j}\omega})$ is the cross-PSD of $y(n)$ and $x(n)$ (see Problem 5.10). This formula holds for both FIR and IIR filters.

If we minimize (5.3.21) with respect to $H(\mathrm{e}^{\mathrm{j}\omega})$, we obtain the system function of the optimum filter and the MMSE. However, we leave this for Problem 5.11 and instead express (5.3.17) in the frequency domain by using (5.3.20). Indeed, we have

$$\begin{aligned} P_o &= r_y(0) - \frac{1}{2\pi}\int_{-\pi}^{\pi} H_o(\mathrm{e}^{\mathrm{j}\omega})R_{yx}^*(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} [R_y(\mathrm{e}^{\mathrm{j}\omega}) - H_o(\mathrm{e}^{\mathrm{j}\omega})R_{yx}^*(\mathrm{e}^{\mathrm{j}\omega})]\,\mathrm{d}\omega \end{aligned} \tag{5.3.22}$$

where $H_o(\mathrm{e}^{\mathrm{j}\omega})$ is the frequency response of the optimum filter. The above equation holds for any filter, FIR or IIR, as long as we use the proper limits to compute the summation in (5.3.19).

We will now obtain a formula for the MMSE that holds only for IIR filters whose impulse response extends from $-\infty$ to ∞. In this case, (5.3.16) is a convolution equation that holds for $-\infty < m < \infty$. Using the convolution theorem of the Fourier transform, we obtain

$$H_o(\mathrm{e}^{\mathrm{j}\omega}) = \frac{R_{yx}(\mathrm{e}^{\mathrm{j}\omega})}{R_x(\mathrm{e}^{\mathrm{j}\omega})} \tag{5.3.23}$$

which, we again stress, holds for noncausal IIR filters *only*. Substituting into (5.4.22), we obtain

$$P_o = \frac{1}{2\pi}\int_{-\pi}^{\pi} [1 - \frac{|R_{yx}(\mathrm{e}^{\mathrm{j}\omega})|^2}{R_y(\mathrm{e}^{\mathrm{j}\omega})R_x(\mathrm{e}^{\mathrm{j}\omega})}]R_y(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega$$

or

$$P_o = \frac{1}{2\pi}\int_{-\pi}^{\pi} [1 - \mathcal{G}_{yx}(\mathrm{e}^{\mathrm{j}\omega})]R_y(\mathrm{e}^{\mathrm{j}\omega})\,\mathrm{d}\omega \tag{5.3.24}$$

where $\mathcal{G}_{yx}(\mathrm{e}^{\mathrm{j}\omega})$ is the coherence function between $x(n)$ and $y(n)$.

This important equation indicates that the performance of the optimum filter depends on the coherence between the input and desired response processes. As we recall from Section 4.4, the coherence is a measure of both the noise disturbing the observations and the relative linearity between $x(n)$ and $y(n)$. The optimum filter can reduce the MMSE at a certain band only if there is significant coherence, that is, $\mathcal{G}_{yx}(\mathrm{e}^{\mathrm{j}\omega}) \simeq 1$. Thus, the optimum filter

$H_o(z)$ constitutes the best, in the MMSE sense, linear relationship between the stochastic processes $x(n)$ and $y(n)$. These interpretations apply to causal IIR and FIR optimum filters, even if (5.4.23) and (5.4.24) only hold approximately in these cases (see Section 5.6).

5.4 Linear Prediction

Linear prediction plays a prominent role in many theoretical, computational, and practical areas of signal processing and deals with the problem of estimating or predicting the value $x(n)$ of a signal at the time instant $n = n_0$, by using a set of other samples from the *same* signal. Although linear prediction is a subject useful in itself, its importance in signal processing is also due, as we will see later, to its use in the development of fast algorithms for optimum filtering and its relation to all-pole signal modeling.

5.4.1 Linear Signal Estimation

Suppose that we are given a set of values $x(n), x(n-1), \cdots, x(n-M)$ of a stochastic process and we wish to estimate the value of $x(n-i)$, using a linear combination of the remaining samples. The resulting estimate and the corresponding estimation error are given by

$$\hat{x}(n-i) \triangleq -\sum_{\substack{k=0 \\ k\neq i}}^{M} c_k^*(n)x(n-k) \tag{5.4.1}$$

and

$$\begin{aligned} e^{(i)}(n) &\triangleq x(n-i) - \hat{x}(n-i) \\ &= \sum_{k=0}^{M} c_k^*(n)x(n-k) \qquad \text{with } c_i(n) \triangleq 1 \end{aligned} \tag{5.4.2}$$

where $c_k(n)$ are the coefficients of the estimator as a function of discrete-time index n. The process is illustrated in Figure 5.16.

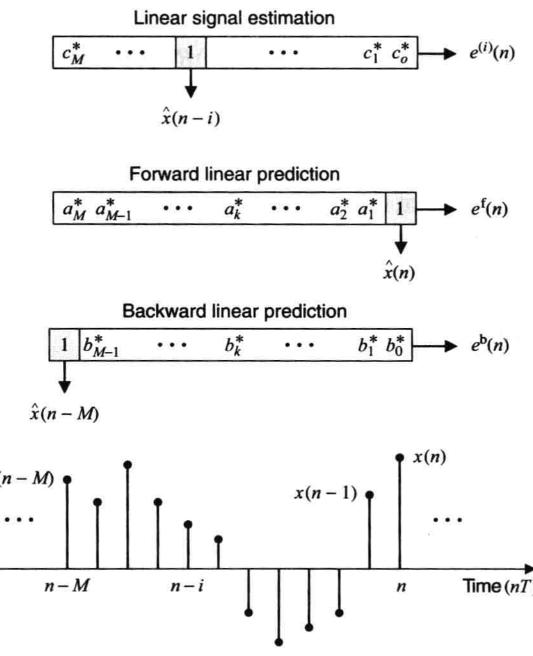


FIGURE 5.16
Illustration showing the samples, estimates, and errors used in linear signal estimation, forward linear prediction, and backward linear prediction.

To determine the MMSE signal estimator, we partition (5.4.2) as

$$\begin{aligned} e^{(i)}(n) &= \sum_{k=0}^{i-1} c_k^*(n)x(n-k) + x(n-i) + \sum_{k=i+1}^{M} c_k^*(n)x(n-k) \\ &\triangleq \boldsymbol{c}_1^{\mathrm{H}}(n)\boldsymbol{x}_1(n) + x(n-i) + \boldsymbol{c}_2^{\mathrm{H}}(n)\boldsymbol{x}_2(n) \\ &\triangleq [\bar{\boldsymbol{c}}^{(i)}(n)]^H \bar{\boldsymbol{x}}(n) \end{aligned} \tag{5.4.3}$$

where the partitions of the coefficient and data vectors, around the ith component, are easily defined from the context. To obtain the normal equations and the MMSE for the optimum *linear signal estimator*, we note that

$$\text{Desired response} = x(n-i) \qquad \text{data vector} = \begin{bmatrix} \boldsymbol{x}_1(n) \\ \boldsymbol{x}_2(n) \end{bmatrix}$$

Using (5.4.6) and (5.4.9) or the orthogonality principle, we have

$$\begin{bmatrix} \boldsymbol{R}_{11}(n) & \boldsymbol{R}_{12}(n) \\ \boldsymbol{R}_{12}^T(n) & \boldsymbol{R}_{22}(n) \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_1(n) \\ \boldsymbol{c}_2(n) \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}_1(n) \\ \boldsymbol{r}_2(n) \end{bmatrix} \tag{5.4.4}$$

or more compactly[4]

$$\boldsymbol{R}^{(i)}(n)\boldsymbol{c}_o^{(i)}(n) = -\boldsymbol{d}^{(i)}(n) \tag{5.4.5}$$

and

$$P_o^{(i)}(n) = P_x(n-i) + \boldsymbol{r}_1^{\mathrm{H}}(n)\boldsymbol{c}_1(n) + \boldsymbol{r}_2^{\mathrm{H}}(n)\boldsymbol{c}_2(n) \tag{5.4.6}$$

Where for, $i, j = 1, 2$

$$\boldsymbol{R}_{ij}(n) \triangleq E\{\boldsymbol{x}_i(n)\boldsymbol{x}_j^{\mathrm{H}}(n)\} \tag{5.4.7}$$

$$\boldsymbol{r}_j(n) \triangleq E\{\boldsymbol{x}_j(n)x^*(n-i)\} \tag{5.4.8}$$

$$P_x(n) = E\{|x(n)|^2\} \tag{5.4.9}$$

For various reasons, to be seen later, we will combine (5.5.4) and (5.5.6) into a single equation. To this end, we note that the correlation matrix of the extended vector

$$\bar{\boldsymbol{x}}(n) = \begin{bmatrix} \boldsymbol{x}_1(n) \\ x(n-i) \\ \boldsymbol{x}_2(n) \end{bmatrix} \tag{5.4.10}$$

can be partitioned as

$$\bar{\boldsymbol{R}}(n) = E\{\bar{\boldsymbol{x}}(n)\bar{\boldsymbol{x}}^{\mathrm{H}}(n)\} = \begin{bmatrix} \boldsymbol{R}_{11}(n) & \boldsymbol{r}_1(n) & \boldsymbol{R}_{12}(n) \\ \boldsymbol{r}_1^{\mathrm{H}}(n) & P_x(n-i) & \boldsymbol{r}_2^{\mathrm{H}}(n) \\ \boldsymbol{R}_{12}^{\mathrm{H}}(n) & \boldsymbol{r}_2(n) & \boldsymbol{R}_{22}(n) \end{bmatrix} \tag{5.4.11}$$

with respect to its ith row and ith column. Using (5.5.4), (5.5.6), and (5.5.11), we obtain

$$\bar{\boldsymbol{R}}(n)\bar{\boldsymbol{c}}_o^{(i)}(n) = \begin{bmatrix} 0 \\ P_o^{(i)}(n) \\ 0 \end{bmatrix} \leftarrow i\text{th row} \tag{5.4.12}$$

which completely determines the linear signal estimator $\mathbf{c}^{(i)}(n)$ and the MMSE $P_o^{(i)}(n)$.

If $M = 2L$ and $i = L$, we have a *symmetric linear smoother* $\bar{\mathbf{c}}(n)$ that produces an estimate of the middle sample by using the L past and the L future samples. The above formulation suggests an easy procedure for the

[4] The minus sign on the right-hand side of the normal equations is the result of arbitrarily setting the coefficient $c_i(n) \triangleq 1$.

computation of the linear signal estimator for any value of i, which is outlined in Table 5.3 and implemented by the function `[ci,Pi]=lsigest(R,i)`. We next discuss two types of linear signal estimation that are of special interest and have their own dedicated notation.

TABLE 5.3
Steps for the computation of optimum signal estimators.

1. Determine the matrix $\bar{R}(n)$ of the extended data vector $\bar{x}(n)$.
2. Create the $M \times M$ submatrix $\mathbf{R}^{(i)}(n)$ of $\bar{\mathbf{R}}(n)$ by removing its ith row and its ith column.
3. Create the $M \times 1$ vector $d^{(i)}(n)$ by extracting the ith column $\bar{d}^{(i)}(n)$ of $\bar{R}(n)$ and removing its ith element.
4. Solve the linear system $R^{(i)}(n)c_o^{(i)}(n) = -d^{(i)}(n)$ to obtain $c_o^{(i)}(n)$.
5. Compute the MMSE $P_o^{(i)}(n) = [\bar{d}^{(i)}(n)]^H \bar{c}_o^{(i)}(n)$.

5.4.2 Forward Linear Prediction

One-step *forward linear prediction* (*FLP*) involves the estimation or prediction of the value $x(n)$ of a stochastic process by using a linear combination of the past samples $x(n-1),\cdots,x(n-M)$ (see Figure 5.16). We should stress that in signal processing applications of linear prediction, what is important is the ability to obtain a good estimate of a sample, pretending that it is unknown, instead of forecasting the future. Thus, the term *prediction* is used more with the signal estimation than forecasting in mind. The forward predictor is a linear signal estimator with $i = 0$ and is denoted by

$$\begin{aligned} e^f(n) &\triangleq x(n) + \sum_{k=1}^{M} a_k^*(n)x(n-k) \\ &= x(n) + \boldsymbol{a}^{\mathrm{H}}(n)\boldsymbol{x}(n-1) \end{aligned} \tag{5.4.13}$$

where

$$\mathbf{a}(n) \triangleq [a_1(n)\ a_2(n)\ \cdots\ a_M(n)]^{\mathrm{T}} \tag{5.4.14}$$

is known as the *forward linear predictor* and $a_k(n)$ with $a_0(n) \triangleq 1$ as the FLP error filter. To obtain the normal equations and the MMSE for the optimum FLP, we note that for $i = 0$, (5.4.11) can be written as

$$\bar{\boldsymbol{R}}(n) = \begin{bmatrix} P_x(n) & \boldsymbol{r}^{f\mathrm{H}}(n) \\ \boldsymbol{r}^f(n) & \boldsymbol{R}(n-1) \end{bmatrix} \tag{5.4.15}$$

where

$$\boldsymbol{R}(n) = E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\} \tag{5.4.16}$$

and

$$\boldsymbol{r}^{\mathrm{f}}(n) = E\{\boldsymbol{x}(n-1)x^*(n)\} \tag{5.4.17}$$

Therefore, (5.4.5) and (5.4.6) give

$$\boldsymbol{R}(n-1)\boldsymbol{a}_o(n) = -\boldsymbol{r}^f(n) \tag{5.4.18}$$

and

$$P_o^f(n) = P_x(n) + \boldsymbol{r}^{f\mathrm{H}}(n)\boldsymbol{a}_o(n) \tag{5.4.19}$$

or

$$\bar{\boldsymbol{R}}(n)\begin{bmatrix} 1 \\ \boldsymbol{a}_o(n) \end{bmatrix} = \begin{bmatrix} P_o^{\mathrm{f}}(n) \\ 0 \end{bmatrix} \tag{5.4.20}$$

which completely specifies the FLP parameters.

5.4.3 Backward Linear Prediction

In this case, we want to estimate the sample $x(n-M)$ in terms of the future samples $x(n), x(n-1),\ldots, x(n-M+1)$

(see Figure 5.16). The term *backward linear prediction* (*BLP*) is not accurate but is used since it is an established convention. A more appropriate name might be *postdiction* or *hindsight*. The BLP is basically a linear signal estimator with $i = M$ and is denoted by

$$\begin{aligned} e^b(n) &\triangleq \sum_{k=0}^{M-1} b_k^*(n)x(n-k) + x(n-M) \\ &= \boldsymbol{b}^H(n)\boldsymbol{x}(n) + x(n-M) \end{aligned} \tag{5.4.21}$$

where

$$b(n) \triangleq [b_0(n)\ b_1(n)\ \cdots\ b_{M-1}(n)]^{\mathrm{T}} \tag{5.4.22}$$

is the BLP and $b_k(n)$ with $b_M(n) \triangleq 1$ is the *backward prediction error filter* (*BPEF*). For $i = M$, (5.4.11) gives

$$\bar{\boldsymbol{R}}(n) = \begin{bmatrix} \boldsymbol{R}(n) & \boldsymbol{r}^b(n) \\ \boldsymbol{r}^{b\mathrm{H}}(n) & P_x(n-M) \end{bmatrix} \tag{5.4.23}$$

where

$$\boldsymbol{r}^{\mathrm{b}}(n) \triangleq E\{\boldsymbol{x}(n)x^*(n-M)\} \tag{5.4.24}$$

The optimum backward linear predictor is specified by

$$\boldsymbol{R}(n)\boldsymbol{b}_o(n) = -\boldsymbol{r}^{\mathrm{b}}(n) \tag{5.4.25}$$

and the MMSE is

$$P_o^b(n) = P_x(n-M) + \boldsymbol{r}^{b\mathrm{H}}(n)\boldsymbol{b}_o(n) \tag{5.4.26}$$

and can be put in a single equation as

$$\bar{\boldsymbol{R}}(n)\begin{bmatrix} \boldsymbol{b}_o(n) \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ P_o^{\mathrm{b}}(n) \end{bmatrix} \tag{5.4.27}$$

In Table 5.4, we summarize the definitions and design equations for optimum FIR filtering and prediction. Using the entries in this table, we can easily obtain the normal equations and the MMSE for the FLP and BLP from those of the optimum filter.

TABLE 5.4
Summary of the design equations for optimum FIR filtering and prediction.

	Optimum filter	FLP	BLP
Input data vector	$\boldsymbol{x}(n)$	$\boldsymbol{x}(n-1)$	$\boldsymbol{x}(n)$
Desired response	$y(n)$	$x(n)$	$x(n-M)$
Coefficient vector	$\boldsymbol{h}(n)$	$\boldsymbol{a}(n)$	$\boldsymbol{b}(n)$
Estimation error	$e(n)=y(n)-\boldsymbol{c}^{\mathrm{H}}(n)\boldsymbol{x}(n)$	$e^{\mathrm{f}}(n) = x(n) - \boldsymbol{a}^{\mathrm{H}}(n)\boldsymbol{x}(n-1)$	$e^{\mathrm{b}}(n) = x(n-M) - \boldsymbol{b}^{\mathrm{H}}(n)\boldsymbol{x}(n)$
Normal equations	$\boldsymbol{R}(n)\boldsymbol{h}_o(n)=\boldsymbol{d}(n)$	$\boldsymbol{R}(n-1)\boldsymbol{a}_o(n) = -\boldsymbol{r}^{\mathrm{f}}(n)$	$\boldsymbol{R}(n)\boldsymbol{b}_o(n) = -\boldsymbol{r}^{\mathrm{b}}(n)$
MMSE	$P_o^{\mathrm{c}}(n) = P_y(n) - \boldsymbol{c}_o^{\mathrm{H}}(n)\boldsymbol{d}(n)$	$P_o^{\mathrm{f}}(n) = P_x(n) + \boldsymbol{a}^{\mathrm{H}}(n)\boldsymbol{r}_o^{\mathrm{f}}(n)$	$P_o^{\mathrm{b}}(n) = P_x(n-M) + \boldsymbol{b}^{\mathrm{H}}(n)\boldsymbol{r}_o^{\mathrm{b}}(n)$
Required moments	$\boldsymbol{R}(n) = E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\}$ $\boldsymbol{d}(n) = E\{\boldsymbol{x}(n)y^*(n)\}$	$\boldsymbol{r}^{\mathrm{f}}(n) = E\{\boldsymbol{x}(n-1)x^*(n)\}$	$\boldsymbol{r}^{\mathrm{b}}(n) = E\{\boldsymbol{x}(n)x^*(n-M)\}$
Stationary processes	$\boldsymbol{R}\boldsymbol{c}_o = \boldsymbol{d}, \boldsymbol{R}$ is Toeplitz	$\boldsymbol{R}\boldsymbol{a}_o = -\boldsymbol{r}^*$	$\boldsymbol{R}\boldsymbol{b}_o = -\boldsymbol{J}\boldsymbol{r} \Rightarrow \boldsymbol{b}_o = \boldsymbol{J}\boldsymbol{a}_o^*$

5.4.4 Stationary Processes

If the process $x(n)$ is stationary, then the correlation matrix $\bar{\boldsymbol{R}}(n)$ does *not* depend on the time n and it is Toeplitz

$$\bar{\boldsymbol{R}} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M) \\ r^*(1) & r(0) & \cdots & r(M-1) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M) & r^*(M-1) & \cdots & r(0) \end{bmatrix} \tag{5.4.28}$$

Therefore, all the resulting linear MMSE signal estimators are *time-invariant*. if we define the correlation vector

$$\boldsymbol{r} \triangleq [r(1)\ r(2)\ \cdots\ r(M)]^{\mathrm{T}} \tag{5.4.29}$$

where $r(l) = E\{x(n)x^*(n-l)\}$, we can easily see that the cross-correlation vectors for the FLP and the BLP are

$$\boldsymbol{r}^f = E\{\boldsymbol{x}(n-1)x^*(n)\} = \boldsymbol{r}^* \tag{5.4.30}$$

and

$$\boldsymbol{r}^{\mathrm{b}} = E\{\boldsymbol{x}(n)x^*(n-M)\} = \boldsymbol{J}\boldsymbol{r} \tag{5.4.31}$$

where

$$\boldsymbol{J} = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{bmatrix} \tag{5.4.32}$$

is the exchange matrix that simply reverses the order of the vector elements. Therefore,

$$\boldsymbol{R}\boldsymbol{a}_o = -\boldsymbol{r}^* \tag{5.4.33}$$

$$P_o^{\mathrm{f}} = r(0) + \boldsymbol{r}^{\mathrm{H}}\boldsymbol{a}_o \tag{5.4.34}$$

$$\boldsymbol{R}\boldsymbol{b}_o = -\boldsymbol{J}\boldsymbol{r} \tag{5.4.35}$$

$$P_o^{\mathrm{b}} = r(0) + \boldsymbol{r}^{\mathrm{H}}\boldsymbol{J}\boldsymbol{b}_o \tag{5.4.36}$$

where the Toeplitz matrix $\boldsymbol{R}$ is obtained from $\bar{\boldsymbol{R}}$ by deleting the last column and row. Using the centrosymmetry property of symmetric Toeplitz matrices

$$\boldsymbol{R}\boldsymbol{J} = \boldsymbol{J}\boldsymbol{R}^* \tag{5.4.37}$$

and (5.5.33), we have

$$\boldsymbol{J}\boldsymbol{R}^*\boldsymbol{a}_o^* = -\boldsymbol{J}\boldsymbol{r} \quad \text{or} \quad \boldsymbol{R}\boldsymbol{J}\boldsymbol{a}_o^* = -\boldsymbol{J}\boldsymbol{r} \tag{5.4.38}$$

Comparing the last equation with (5.5.35), we have

$$\boldsymbol{b}_o = \boldsymbol{J}\boldsymbol{a}_o^* \tag{5.4.39}$$

that is, the BLP coefficient vector is the reverse of the conjugated FLP coefficient vector. Furthermore, from (5.4.34), (5.4.35), and (5.4.39), we have

$$P_o \triangleq P_o^{\mathrm{f}} = P_o^{\mathrm{b}} \tag{5.4.40}$$

that is, the forward and backward prediction error powers are equal.

This remarkable symmetry between the MMSE forward and backward linear predictors holds for stationary processes but disappears for nonstationary processes. Also, we do not have such a symmetry if a criterion other than the MMSE is used and the process to be predicted is non-Gaussian (Weiss 1975; Lawrence 1991).

EXAMPLE 5.4.1. To illustrate the basic ideas in FLP, BLP, and linear smoothing, we consider the second-order estimators for stationary processes.

The augmented equations for the first-order FLP are

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} P_1^f \\ 0 \end{bmatrix}$$

and they can be solved by using Cramer's rule. Indeed, we obtain

$$a_0^{(1)} = \frac{\det \begin{bmatrix} P_1^f & r(1) \\ 0 & r(0) \end{bmatrix}}{\det \boldsymbol{R}_2} = \frac{r(0)P_1^f}{\det \boldsymbol{R}_2} = 1 \Rightarrow P_1^f = \frac{\det \boldsymbol{R}_2}{\det \boldsymbol{R}_1} = \frac{r^2(0) - |r(1)|^2}{r(0)}$$

and

$$a_1^{(1)} = \frac{\det \begin{bmatrix} r(0) & P_1^f \\ r^*(1) & 0 \end{bmatrix}}{\det \boldsymbol{R}_2} = \frac{-P_1^f r^*(1)}{\det \boldsymbol{R}_2} = -\frac{r^*(1)}{r(0)}$$

for the MMSE and the FLP. For the second-order case we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} P_2^f \\ 0 \\ 0 \end{bmatrix}$$

whose solution is

$$a_0^{(2)} = \frac{P_2^f \det \boldsymbol{R}_2}{\det \boldsymbol{R}_3} = 1 \Rightarrow P_2^f = \frac{\det \boldsymbol{R}_3}{\det \boldsymbol{R}_2}$$

$$a_1^{(2)} = \frac{-P_2^f \det \begin{bmatrix} r^*(1) & r(1) \\ r^*(2) & r(0) \end{bmatrix}}{\det \boldsymbol{R}_3} = \frac{-\det \begin{bmatrix} r^*(1) & r(1) \\ r^*(2) & r(0) \end{bmatrix}}{\det \boldsymbol{R}_2} = \frac{r(1)r^*(2) - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

and

$$a_2^{(2)} = \frac{P_2^f \det \begin{bmatrix} r^*(1) & r(0) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \boldsymbol{R}_3} = \frac{\det \begin{bmatrix} r^*(1) & r(0) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \boldsymbol{R}_2} = \frac{[r^*(1)]^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}$$

Similarly, for the BLP

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ P_1^b \end{bmatrix}$$

where $b_1^{(1)} = 1$, we obtain

$$P_1^b = \frac{\det \boldsymbol{R}_2}{\det \boldsymbol{R}_1} \quad \text{and} \quad b_0^{(1)} = -\frac{r(1)}{r(0)}$$

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} b_0^{(2)} \\ b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ P_2^b \end{bmatrix}$$

$$P_2^b = \frac{\det \boldsymbol{R}_3}{\det \boldsymbol{R}_2} \qquad b_1^{(2)} = \frac{r^*(1)r(2) - r(0)r(1)}{r^2(0) - |r(1)|^2} \qquad b_0^{(2)} = \frac{r^2(1) - r(0)r(2)}{r^2(0) - |r(1)|^2}$$

We note that

$$P_1^f = P_1^b \qquad a_1^{(1)} = b_1^{(1)*}$$

and

$$P_2^f = P_2^b \qquad a_1^{(2)} = b_1^{(2)*} \qquad a_2^{(2)} = b_0^{(2)*}$$

which is a result of the stationarity of $x(n)$ or equivalently of the Toeplitz structure of R_m.

For the linear signal estimator, we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} c_0^{(2)} \\ c_1^{(2)} \\ c_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ P_2 \\ 0 \end{bmatrix}$$

with $c_1^{(2)} = 1$. Using Cramer's rule, we obtain

$$P_2 = \frac{\det \boldsymbol{R}_3}{\det \boldsymbol{R}_3^{(2)}}$$

$$c_0^{(2)} = \frac{-P_2 \det \begin{bmatrix} r(1) & r(2) \\ r^*(1) & r(0) \end{bmatrix}}{\det \boldsymbol{R}_3} = -\frac{\det \begin{bmatrix} r(1) & r(2) \\ r^*(1) & r(0) \end{bmatrix}}{\det \boldsymbol{R}_3^{(2)}} = \frac{r^*(1)r(2) - r(0)r(1)}{r^2(0) - |r(1)|^2}$$

$$c_2^{(2)} = \frac{-P_2 \det \begin{bmatrix} r(0) & r(1) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \boldsymbol{R}_3} = -\frac{\det \begin{bmatrix} r(0) & r(1) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \boldsymbol{R}_3^{(2)}} = \frac{r(1)r^*(2) - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

from which we see that $c_0^{(2)} = c_2^{(2)*}$; that is, we have a linear phase estimator.

5.4.5 Properties

Linear signal estimators and predictors have some interesting properties that we discuss next.

PROPERTY 5.4.1. If the process $x(n)$ is stationary, then the symmetric, linear smoother has linear phase.

Proof. Using the centrosymmetry property $\bar{\boldsymbol{R}}\boldsymbol{J} = \boldsymbol{J}\bar{\boldsymbol{R}}^*$ and (5.5.12) for $M = 2L$, $i = L$, we obtain

$$\bar{\boldsymbol{c}} = \boldsymbol{J}\bar{\boldsymbol{c}}^* \tag{5.4.41}$$

that is, the symmetric, linear smoother has even symmetry and, therefore, has linear phase (see Problem 5.12).

PROPERYT 5.4.2. If the process $x(n)$ is stationary, the forward prediction error filter (PEF) $1, a_1, a_2, \cdots, a_M$ is minimum-phase and the backward PEF $b_0, b_1, \cdots, b_{M-1}, 1$ is maximum-phase.

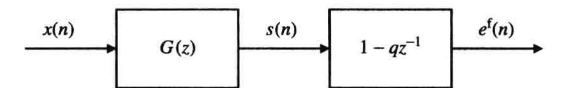


FIGURE 5.17
The prediction error filter with one zero factored out.

Proof. The system function of the M th-order forward PEF can be factored as

$$A(z) = 1 + \sum_{k=1}^{M} a_k^* z^{-k} = G(z)(1 - qz^{-1})$$

where q is a zero of $A(z)$ and

$$G(z) = 1 + \sum_{k=1}^{M-1} g_k z^{-k}$$

is an $(M-1)$ st-order filter. The filter $A(z)$ can be implemented as the cascade connection of the filters $G(z)$ and $1 - qz^{-1}$ (see Figure 5.17). The output $s(n)$ of $G(z)$ is

$$s(n) = x(n) + g_1 x(n-1) + \cdots + g_{M-1} x(n - M + 1)$$

and it is easy to see that

$$E\{s(n-1)e^{f*}(n)\} = 0 \tag{5.4.42}$$

because $E\{x(n-k)e^{f*}(n)\} = 0$ for $1 \le k \le M$. Since the output of the second filter can be expressed as

$$e^f(n) = s(n) - qs(n-1)$$

we have

$$E\{s(n-1)e^{f*}(n)\} = E\{s(n-1)s^*(n)\} - q^* E\{s(n-1)s^*(n-1)\} = 0$$

which implies that

$$q = \frac{r_s(-1)}{r_s(0)} \Rightarrow |q| \le 1$$

because q is equal to the normalized autocorrelation of $s(n)$. If the process $x(n)$ is not predictable, that is, $E\{|e^f(n)|^2\} \ne 0$, we have

$$\begin{aligned} E\{|e^{f}(n)|^2\} &= E\{e^{f}(n)[s^*(n)-q^*s^*(n-1)]\} \\ &= E\{e^{f}(n)s^*(n)\} \qquad \text{due to} \\ &= E\{[s(n)-qs(n-1)s^*(n)]\} \\ &= r_s(0)(1-|q|^2)\neq 0 \end{aligned} \tag{5.4.42}$$

which implies that $|q| < 1$

that is, the zero q of the forward PEF filter is strictly inside the unit circle. Repeating this process, we can show that all zeros of A(z) are inside the unit circle; that is, A(z) is minimum-phase. This proof was presented in Vaidyanathan et al. (1996). The property ***b=Ja**** is equivalent to

$$B(z) = z^{-M}A^*\left(\frac{1}{z^*}\right)$$

which implies that $B(z)$ is a maximum-phase filter.

PROPERTY 5.4.3. The forward and backward prediction error filters can be expressed in terms of the eigenvalues $\bar{\lambda}_i$ and the eigenvectors $\bar{\boldsymbol{q}}_i$ of the correlation matrix $\bar{\mathbf{R}}(n)$ as follows

$$\begin{bmatrix} 1 \\ \boldsymbol{a}_o(n) \end{bmatrix} = P_o^f(n)\sum_{i=1}^{M+1}\frac{1}{\bar{\lambda}_i}\bar{\boldsymbol{q}}_i\bar{q}_{i,1}^* \tag{5.4.43}$$

and

$$\begin{bmatrix} \boldsymbol{b}_o(n) \\ 1 \end{bmatrix} = P_o^b(n)\sum_{i=1}^{M+1}\frac{1}{\bar{\lambda}_i}\bar{\boldsymbol{q}}_i\bar{q}_{i,M+1}^* \tag{5.4.44}$$

where $\bar{q}_{i,1}$ and $\bar{q}_{i,M+1}$ are the first and last components of $\bar{q}_i$. The first equation of (5.4.43) and the last equation in (5.4.44) can be solved to provide the MMSEs $P_o^{\mathrm{f}}(n)$ and $P_o^{\mathrm{b}}(n)$, respectively.

Proof. See Problem 5.14.

PROPERTY 5.4.4. The MMSE prediction errors can be expressd as

$$P_o^{\mathrm{f}}(n) = \frac{\det\bar{\boldsymbol{R}}(n)}{\det\boldsymbol{R}(n-1)} \qquad P_o^b(n) = \frac{\det\bar{\boldsymbol{R}}(n)}{\det\boldsymbol{R}(n)} \tag{5.4.45}$$

Proof. Problem 5.17.

The previous concepts are illustrated in the following example.

EXAMPLE 5.4.2. A random sequence $x(n)$ is generated by passing the white Gaussian noise process $\omega(n)\sim \mathrm{WN}(0,1)$ through the filter

$$x(n) = \omega(n) + \frac{1}{2}\omega(n-1)$$

Determine the second-order FLP, BLP, and symmetric linear signal smoother.

Solution. The complex power spectrum is

$$R(z) = H(z)H(z^{-1}) = (1+\frac{1}{2}z^{-1})(1+\frac{1}{2}z) = \frac{1}{2}z + \frac{5}{4} + \frac{1}{2}z^{-1}$$

Therefore, the autocorrelation sequence is equal to $r(0)=5/4,\ r(\pm1)=1/2,\ r(l)=0$ for $|l|\geq 2$. Since the power spectrum $R(e^{j\omega}) = 5/4+\cos\omega > 0$ for all ω, the autocorrelation matrix is positive definite. The same is true of any principal submatrix. To determine the second-order linear signal estimators, we start with the matrix

$$\bar{\boldsymbol{R}} = \begin{bmatrix} \frac{5}{4} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{5}{4} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{5}{4} \end{bmatrix}$$

and follow the procedure outlined in Section 5.4.1 or use the formulas in Table 5.3. The results are

Forward linear prediction ($i=0$): $\{a_k\} \to \{1, \ -0.476, \ 0.190\}$ $P_o^f = 1.0119$

Symmetric linear smoothing ($i=1$): $\{c_k\} \to \{-0.4, \ 1, \ -0.4\}$ $P_o^s = 0.8500$

Backward linear prediction ($i=2$): $\{b_k\} \to \{0.190, \ -0.476, \ 1\}$ $P_o^b = 1.0119$

The inverse of the correlation matrix $\bar{R}$ is

$$\bar{R}^{-1} = \begin{bmatrix} 0.9882 & -0.4706 & 0.1882 \\ -0.4706 & 1.1765 & -0.4706 \\ 0.1882 & -0.4706 & 0.9882 \end{bmatrix}$$

and we see that dividing the first, second, and third columns by 0.9882, 1.1765, and 0.9882 provides the forward PEF, the symmetric linear smoothing filter, and the backward PEF, respectively. The inverses of the diagonal elements provide the MMSEs P_o^f, P_o^s, and P_o^b. The reader can easily see, by computing the zeros of the corresponding system functions, that the FLP is minimum-phase, the BLP is maximum-phase, and the symmetric linear smoother is mixed-phase. It is interesting to note that the smoother performs better than either of the predictors.

5.5 Optimum Infinite Impulse Response Filters

So far we have dealt with optimum FIR filters and predictors for nonstationary and stationary processes. In this section, we consider the design of optimum IIR filters for stationary stochastic processes. For nonstationary processes, the theory becomes very complicated. The Wiener-Hopf equations for optimum IIR filters are the same for FIR filters; only the limits in the convolution summation and the range of values for which the normal equations hold are different. Both are determined by the limits of summation in the filter convolution equation. We can easily see from (5.3.16) and (5.3.17), or by applying the orthogonality principle (5.2.41), that the optimum IIR filter

$$\hat{y}(n) = \sum_k h_o(k)x(n-k) \tag{5.5.1}$$

is specified by the Wiener-Hopf equations

$$\sum_k h_o(k)r_x(m-k) = r_{yx}(m) \tag{5.5.2}$$

and the MMSE is given by

$$P_o = r_y(0) - \sum_k h_o(k)r_{yx}^*(k) \tag{5.5.3}$$

where $r_x(l)$ is the autocorrelation of the input stochastic process $x(n)$ and $r_{yx}(l)$ is the cross-correlation between $x(n)$ and desired response process $y(n)$. We assume that the processes $x(n)$ and $y(n)$ are jointly wide-sense stationary with zero mean values.

The range of summation in the above equations includes all the nonzero coefficients of the impulse response of the filter. The range of k in (5.5.1) determines the number of unknowns and the number of equations, that is, the range of m. For IIR filters, we have an infinite number of equations and unknowns, and thus only analytical solutions for (5.5.2) are possible. The key to analytical solutions is that the left-hand side of (5.5.2) can be expressed as the convolution of $h_o(m)$ with $r_x(m)$, that is,

$$h_o(m) * r_x(m) = r_{yx}(m) \tag{5.5.4}$$

which is a *convolutional equation* that can be solved by using the z-transform. The complexity of the solution depends on the range of m.

The formula for the MMSE is the same for any filter, either FIR or IIR. Indeed, using Parseval's theorem and (5.5.3), we obtain

$$P_o = r_y(0) - \frac{1}{2\pi j} \oint_C H_o(z)R_{yx}^*\left(\frac{1}{z^*}\right) z^{-1}\,dz \tag{5.5.5}$$

where $H_o(z)$ is the system function of the optimum filter and $R_{yx}(z) = Z\{r_{yx}(l)\}$. The power P_y can be computed by

$$P_o = r_y(0) - \frac{1}{2\pi j} \oint_C R_y(z)z^{-1}\,dz \tag{5.5.6}$$

where $R_y(z) = Z\{r_y(l)\}$. Combining (5.5.5) with (5.5.6), we obtain

$$P_o = \frac{1}{2\pi j} \oint_C [R_y(z) - H_o(z) R_{yx}^*\left(\frac{1}{z^*}\right)] z^{-1}\, dz \tag{5.5.7}$$

which expresses the MMSE in terms of z-transforms. To obtain the MMSE in the frequency domain, we replace z by $e^{j\omega}$. For example, (5.5.5) becomes

$$P_o = r_y(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} H_o(e^{j\omega}) R_{yx}^*(e^{j\omega})\, d\omega$$

where $H_o(e^{j\omega})$ is the frequency response of the optimum filter.

5.5.1 Noncausal IIR Filters

For the noncausal IIR filter

$$\hat{y}(n) = \sum_{k=-\infty}^{\infty} h_{nc}(k) x(n-k) \tag{5.5.8}$$

the range of the Wiener-Hopf equations (5.5.2) is $-\infty < m < \infty$ and can be easily solved by using the convolution property of the z-transform. This gives

$$H_{nc}(z) R_x(z) = R_{yx}(z)$$

or

$$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)} \tag{5.5.9}$$

where $H_{nc}(z)$ is the system function of the optimum filter, $R_x(z)$ is the complex PSD of $x(n)$, and $R_{yx}(z)$ is the complex cross-PSD between $y(n)$ and $x(n)$.

5.5.2 Causal IIR Filters

For the causal IIR filter

$$\hat{y}(n) = \sum_{k=0}^{\infty} h_c(k) x(n-k) \tag{5.5.10}$$

the Wiener-Hopf equations (5.5.2) hold only for m in the range $0 \le m < \infty$. Since the sequence $r_y(m)$ can be expressed as the convolution of $h_o(m)$ and $r_x(m)$ *only* for $m \ge 0$, we cannot solve (5.5.2) using the z-transform. However, a simple solution is possible using the spectral factorization theorem.[1] This approach was introduced for continuous-time processes in Bode and Shannon (1950) and Zadeh and Ragazzini (1950). It is based on the following two observations:

1. The solution of the Wiener-Hopf equations is trivial if the input is white.
2. Any regular process can be transformed to an equivalent white process.

White input processes. We first note that if the process $x(n)$ is white noise, the solution of the Wiener-Hopf equations is trivial. Indeed, if

$$r_x(l) = \sigma_x^2 \delta(l)$$

Then Equation (5.5.4) gives

$$h_c(m) * \delta(m) = \frac{r_{yx}(m)}{\sigma_x^2} \qquad 0 \le m < \infty$$

[1] An analogous matrix-based approach is extensively used in Chapter 6 for the design and implementation of optimum FIR filters.

which implies that

$$h_c(m) = \begin{cases} \dfrac{1}{\sigma_x^2} r_{yx}(m) & 0 \le m < \infty \\ 0 & m < 0 \end{cases} \tag{5.5.11}$$

because the filter is causal. The system function of the optimum filter is given by

$$H_c(z) = \frac{1}{\sigma_x^2}[R_{yx}(z)]_+ \tag{5.5.12}$$

where

$$[R_{yx}(z)]_+ \triangleq \sum_{l=0}^{\infty} r_{yx}(l) z^{-l} \tag{5.5.13}$$

is the one-sided z-transform of the two-sided sequence $r_{yx}(l)$. The MMSE is given by

$$P_c = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=0}^{\infty} |r_{yx}(k)|^2 \tag{5.5.14}$$

which follows from (5.5.3) and (5.5.11).

Regular input processes. The PSD of a regular process can be factored as

$$R_x(z) = \sigma_x^2 H_x(z) H_x^*\left(\frac{1}{z^*}\right) \tag{5.5.15}$$

where $H_x(z)$ is the innovations filter (see Section 3.1). The innovations process

$$\omega(n) = x(n) - \sum_{k=1}^{\infty} h_x(k) \omega(n-k) \tag{5.5.16}$$

is white and *linearly equivalent* to the input process $x(n)$. Therefore, linear estimation of $y(n)$ based on $x(n)$ is equivalent to linear estimation of $y(n)$ based on $\omega(n)$. The optimum filter that estimates $y(n)$ from $x(n)$ is obtained by cascading the whitening filter $1/H_x(z)$ with the optimum filter that estimates $y(n)$ from $\omega(n)$ (see Figure 5.18). Since $\omega(n)$ is white, the optimum filter for estimating $y(n)$ from $\omega(n)$ is

$$H_c'(z) = \frac{1}{\sigma_x^2}[R_{y\omega}(z)]_+ \tag{5.5.17}$$

where $[R_{y\omega}(z)]_+$ is the one-sided z-transform of $r_{y\omega}(l)$. To express $H_c'(z)$ in terms of $R_{yx}(z)$, we need the relationship between $R_{y\omega}(z)$ and $R_{yx}(z)$. From

$$x(n) = \sum_{k=0}^{\infty} h_x(k) \omega(n-k)$$

we obtain

$$E\{y(n)x^*(n-l)\} = \sum_{k=0}^{\infty} h_x^*(k) E\{y(n)\omega^*(n-l-k)\}$$

or

$$r_{yx}(l) = \sum_{k=0}^{\infty} h_x^*(k) r_{y\omega}(l+k) \tag{5.5.18}$$

Taking the z-transform of the above equation leads to

$$R_{y\omega}(z) = \frac{R_{yx}(z)}{H_x^*(1/z^*)} \tag{5.5.19}$$

which, combined with (5.5.17), gives

$$H_c'(z) = \frac{1}{\sigma_x^2}\left[\frac{R_{yx}(z)}{H_x^*(1/z^*)}\right]_+ \tag{5.5.20}$$

which is the causal optimum filter for the estimation of $y(n)$ from $\omega(n)$. The optimum filter for estimating $y(n)$ from $x(n)$ is

$$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)}\left[\frac{R_{yx}(z)}{H_x^*(1/z^*)}\right] \tag{5.5.21}$$

which is causal since it is the cascade connection of two causal filters [see Figure 5.19(*a*)].

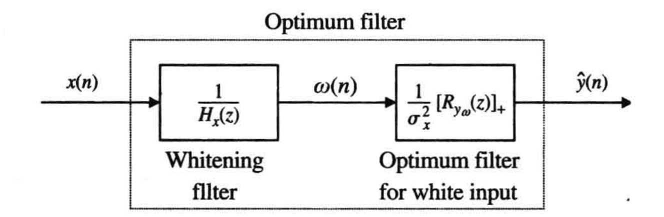
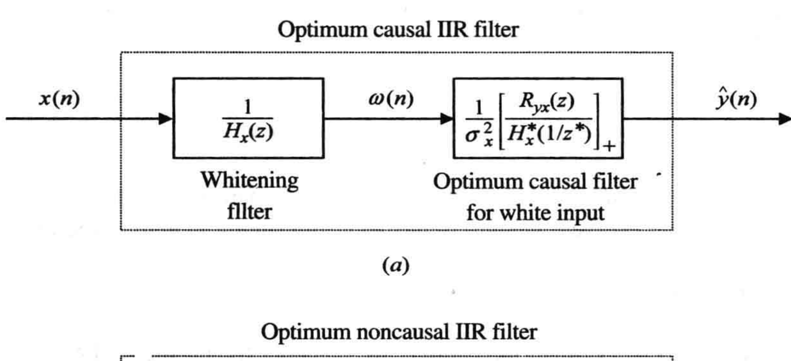


FIGURE 5.18
Optimum causal IIR filter design by the spectral factorization method.



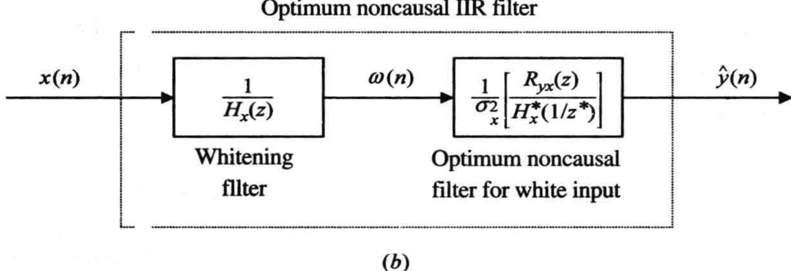


FIGURE 5.19
Comparison of causal and noncausal IIR optimum filters.

The MMSE from (5.5.3) can also be expressed as

$$P_c = r_y(0) - \frac{1}{\sigma_x^2}\sum_{k=0}^{\infty}|r_{y\omega}(k)|^2 \tag{5.5.22}$$

which shows that the MMSE decreases as we increase the order of the filter. Table 5.5 summarizes the equations required for the design of optimum FIR and IIR filters.

TABLE 5.5
Design of FIR and IIR optimum filters for stationary processes.

Filter type	Solution	Required quantities
FIR	$e(n) = y(n) - c_o^H x(n)$ $c_o = R^{-1}d$ $P_o = r_y(0) - d^H c_o$	$R = [r_x(m-k)]$, $d = [r_{yx}(m)]$ $0 \le k,\ m \le M-1,\ M = \text{finite}$
Noncausal IIR	$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)}$ $P_{nc} = r_y(0) - \sum_{k=-\infty}^{\infty} h_{nc}(k) r_{yx}^*(k)$	$R_x(z) = Z\{r_x(l)\}$ $R_{yx}(z) = Z\{r_{xy}^*(l)\}$
Causal IIR	$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_{yx}(z)}{H_x^*(1/z^*)} \right]_+$ $P_c = r_y(0) - \sum_{k=0}^{\infty} h_{nc}(k) r_{yx}^*(k)$	$R_x(z) = \sigma_x^2 H_x(z) H_x^*(1/z^*)$ $R_{yx}(z) = Z\{r_{xy}(l)\}$

Finally, since the equation for the noncausal IIR filter can be written as

$$H_{nc}(z) = \frac{1}{\sigma_x^2 H_x(z)} \frac{R_{yx}(z)}{H_x^*(1/z^*)} \tag{5.5.23}$$

we see that the only difference from the causal filter is that the noncausal filter includes both the causal and noncausal parts of $R_{yx}(z)/H_x(z^{-1})$ [see Figure 5.19(*b*)]. By using the innovations process $\omega(n)$, the MMSE can be expressed as

$$P_{nc} = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=-\infty}^{\infty} |r_{y\omega}(k)|^2 \tag{5.5.24}$$

and is known as the *irreducible MMSE* because it is the best performance that can be achieved by a linear filter. Indeed, since $|r_{y\omega}(k)| \ge 0$, every coefficient we add to the optimum filter can help to reduce the MMSE.

5.5.3 Filtering of Additive Noise

To illustrate the optimum filtering theory developed above, we consider the problem of estimating a "useful" or desired signal $y(n)$ that is corrupted by additive noise $v(n)$. The goal is to find an optimum filter that extracts the signal $y(n)$ from the noisy observations

$$x(n) = y(n) + v(n) \tag{5.5.25}$$

given that $y(n)$ and $v(n)$ are uncorrelated processes with known autocorrelation sequences $r_y(l)$ and $r_v(l)$.

To design the optimum filter, we need the autocorrelation $r_x(l)$ of the input signal $x(n)$ and the cross-correlation $r_{yx}(l)$ between the desired response $y(n)$ and the input signal $x(n)$. Using (5.5.25), we find

$$r_x(l) = E\{x(n)x^*(n-l)\} = r_y(l) + r_v(l) \tag{5.5.26}$$

and

$$r_{yx}(l) = E\{y(n)x^*(n-l)\} = r_y(l) \tag{5.5.27}$$

because $y(n)$ and $v(n)$ are uncorrelated.

The design of optimum IIR filters requires the functions $R_x(z)$ and $R_{yx}(z)$. Taking the z-transform of (5.5.26) and (5.5.27), we obtain

$$R_x(z) = R_y(z) + R_v(z) \tag{5.5.28}$$

and

$$R_{yx}(z) = R_y(z) \tag{5.5.29}$$

The noncausal optimum filter is gien by

$$H_{nc}(z)=\frac{R_{yx}(z)}{R_x(z)}=\frac{R_y(z)}{R_y(z)+R_v(z)} \tag{5.5.30}$$

which for $z=e^{j\omega}$ shows that, for those values of ω for which $|R_y(e^{j\omega})| \gg |R_v(e^{j\omega})|$, that is, for high SNR, we have $|H_{nc}(e^{j\omega})| \approx 1$. In contrast, if $|R_y(e^{j\omega})| \ll |R_v(e^{j\omega})|$, that is, for low SNR, we have $|H_{nc}(e^{j\omega})| \approx 0$. Thus, the optimum filter "passes" its input in bands with high SNR and attenuates it in bands with low SNR, as we would expect intuitively.

Substituting (5.6.30) into (5.6.7), we obtain

$$P_{nc}=\frac{1}{2\pi j}\oint_C \frac{R_y(z)R_v(z)}{R_y(z)+R_v(z)}z^{-1}dz \tag{5.5.31}$$

which provides an expression for the MMSE that does not require knowledge of the optimum filter.

We next illustrate the design of optimum filters for the reduction of additive noise with a detailed numerical example.

EXAMPLE 5.5.1. In this example we illustrate the design of an optimum IIR filter to extract a random signal with known autocorrelation sequence

$$r_y(l)=\alpha^{|l|} \qquad -1<\alpha<1 \tag{5.5.32}$$

which is corrupted by additive white noise with autocorrelation

$$r_v(l)=\sigma_v^2\delta(l) \tag{5.5.33}$$

The processes $y(n)$ and $v(n)$ are uncorrelated.

Required statistical moments. The input to the filter is the signal $x(n)=y(n)+v(n)$ and the desired response, the signal $y(n)$. The first step in the design is to determine the required second-order moments, that is, the autocorrelation of the input process and the cross-correlation between input and desired response. Substituting into (5.5.26) and (5.5.27), we have

$$r_x(l)=\alpha^{|l|}+\sigma_v^2\delta(l) \tag{5.5.34}$$

and

$$r_{yx}(l)=\alpha^{|l|} \tag{5.5.35}$$

To simplify the derivations and deal with "nice, round" numbers, we choose $\alpha=0.8$ and $\sigma_v^2=1$. Then the complex power spectral densities of $y(n)$, $v(n)$, and $x(n)$ are

$$R_y(z)=\frac{\left(\frac{3}{5}\right)^2}{(1-\frac{4}{5}z^{-1})(1-\frac{4}{5}z)} \qquad \frac{4}{5}<|z|<\frac{5}{4} \tag{5.5.36}$$

$$R_v(z)=\sigma_v^2=1 \tag{5.5.37}$$

and

$$R_x(z)=\frac{8}{5}\frac{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)}{(1-\frac{4}{5}z^{-1})(1-\frac{4}{5}z)} \tag{5.5.38}$$

respectively.

Noncausal filter. Using (5.5.9), (5.5.29), (5.5.36), and (5.5.38), we obtain

$$H_{nc}(z)=\frac{R_{yx}(z)}{R_x(z)}=\frac{9}{40}\frac{1}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)} \qquad \frac{1}{2}<|z|<2$$

Evaluating the inverse the z - transform we have

$$h_{\text{nc}}(n) = \frac{3}{10}(\frac{1}{2})^{|n|} \qquad -\infty < n < \infty$$

which clearly corresponds to a noncausal filter. From (5.5.3), the MMSE is

$$P_{\text{nc}} = 1 - \frac{3}{10}\sum_{k=-\infty}^{\infty}(\frac{1}{2})^{|k|}(\frac{4}{5})^{|k|} = \frac{3}{10} \tag{5.5.39}$$

and provides the irreducible MMSE.

Causal filter. To find the optimum causal filter, we need to perform the spectral factorization

$$R_x(z) = \sigma_x^2 H_x(z) H_x(z^{-1})$$

which is provided by (5.5.38) with

$$\sigma_x^2 = \frac{8}{5} \tag{5.5.40}$$

and

$$H_x(z) = \frac{1 - \frac{1}{2}z^{-1}}{1 - \frac{4}{5}z^{-1}} \tag{5.5.41}$$

Thus,

$$R_{yw}(z) = \frac{R_{yx}(z)}{H_x(z^{-1})} = \frac{0.36}{(1 - \frac{4}{5}z^{-1})(1 - \frac{1}{2}z)} = \frac{0.6}{1 - \frac{4}{5}z^{-1}} + \frac{0.3z}{1 - \frac{1}{2}z} \tag{5.5.42}$$

where the first term (causal) converges for $|z| > 4/5$ and the second term (noncausal) converges for $|z| < 2$. Hence, taking the causal part

$$\left[\frac{R_{yx}(z)}{H_x(z^{-1})}\right]_+ = \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}}$$

and substituting into (5.5.21), we obtain the causal optimum filter

$$H_c(z) = \frac{5}{8}\left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}}\right) = \frac{3}{8}\left(\frac{1}{1 - \frac{1}{2}z^{-1}}\right) \qquad |z| < \frac{1}{2} \tag{5.5.43}$$

The impulse response is

$$h_c(n) = \frac{3}{8}(\frac{1}{2})^n u(n)$$

which corresponds to a causal and stable IIR filter. The MMSE is

$$P_c = r_y(0) - \sum_{k=0}^{\infty} h_c(k) r_{yx}(k) = 1 - \frac{3}{8}\sum_{k=0}^{\infty}(\frac{1}{2})^k(\frac{4}{5})^k = \frac{3}{8} \tag{5.5.44}$$

which is, as expected, larger than P_{nc}.

From (5.5.54), we see that the optimum causal filter is a first-order recursive filter that can be implemented by the difference equation

$$\hat{y}(n) = \frac{1}{2}\hat{y}(n-1) + \frac{3}{8}x(n)$$

In general, this is possible only when $H_c(z)$ is a rational function.

Computation of MMSE using the innovation. We next illustrate how to find the MMSE by using the cross-correlation sequence $r_{yw}(l)$. From (5.5.42), we obtain

$$r_{y\omega}(l) = \begin{cases} \frac{3}{5}(\frac{4}{5})^l & l \geq 0 \\ \frac{3}{5}2^l & l < 0 \end{cases} \tag{5.5.45}$$

which, in conjunction with (5.5.22) and (5.5.24), gives

$$P_c = r_y(0) - \frac{1}{\sigma_x^2}\sum_{k=0}^{\infty} r_{y\omega}^2(k) = 1 - \frac{5}{8}(\frac{3}{5})^2 \sum_{k=0}^{\infty}(\frac{4}{5})^{2k} = \frac{3}{8}$$

and

$$P_{nc} = r_y(0) - \frac{1}{\sigma_x^2}\left[\sum_{k=0}^{\infty} r_{y\omega}^2(k) - \sum_{k=-\infty}^{-1} r_{y\omega}^2(k)\right] = \frac{3}{10}$$

which agree with (5.5.44) and (5.5.39).

Noncausal smoothing filter. Suppose now that we want to estimate the value $y(n+D)$ of the desired response from the data $x(n)$, $-\infty < n < \infty$. Since

$$E\{y(n+D)x(n-l)\} = r_{yx}(n+D) \tag{5.5.46}$$

and

$$Z\{r_{yx}(n+D)\} = z^D R_{yx}(z) \tag{5.5.47}$$

the noncausal Wiener smoothing filter is

$$H_{nc}^D(z) = \frac{z^D R_{yx}(z)}{R_x(z)} = \frac{z^D R_y(z)}{R_x(z)} = z^D H_{nc}(z) \tag{5.5.48}$$

$$h_{nc}^D(n) = h_{nc}(n+D) \tag{5.5.49}$$

The MMSE is

$$P_{nc}^D = r_y(0) - \sum_{k=-\infty}^{\infty} h_{nc}(k+D) r_{yx}(k+D) = P_{nc} \tag{5.5.50}$$

which is independent of the time shift D.

Causal prediction filter. We estimate the value $y(n+D)(D>0)$ of the desired response using the data $x(k)$, $-\infty < k \leq n$. The whitening part of the causal prediction filter does not depend on $y(n)$ and is still given by (5.5.41). The coloring part depends on $y(n+D)$ and is given by $R'_{y\omega}(z) = z^D R_{y\omega}(z)$ or $r'_{y\omega}(l) = r_{y\omega}(l+D)$. Taking into consideration that $D>0$, we can show (see Problem 5.31) that the system function and the impulse response of the causal Wiener predictor are

$$H_c^{[D]}(z) = \frac{5}{8}\left(\frac{1-\frac{4}{5}z^{-1}}{1-\frac{1}{2}z^{-1}}\right)\left[\frac{\frac{3}{5}\left(\frac{4}{5}\right)^D}{1-\frac{4}{5}z^{-1}}\right] = \frac{\frac{3}{8}\left(\frac{4}{5}\right)^D}{1-\frac{1}{2}z^{-1}} \tag{5.5.51}$$

and

$$h_c^{[D]}(n) = \frac{3}{8}(\frac{4}{5})^D(\frac{1}{2})^n u(n) \tag{5.5.52}$$

repectively. This shows that as $D \to \infty$, the impulse response $h_c^{[D]}(n) \to 0$, which is consistent with our intuition that the prediction is less and less reliable. The MMSE is

$$P_c^{[D]} = 1 - \frac{3}{8}(\frac{4}{5})^{2D}\sum_{k=0}^{\infty}(\frac{2}{5})^k = 1 - \frac{5}{8}(\frac{4}{5})^{2D} \tag{5.5.53}$$

and $P_c^{[D]} \to r_y(0) = 1$ as $D \to \infty$, which agrees with our earlier observation. For $D=2$, the MMSE is $P_c^{[2]} = 93/125 = 0.7440 > P_c$, as expected.

Causal smoothing filter. To estimate the value $y(n+D)$ $(D<0)$ of the desired response using the data $x(n)$, $-\infty < k \leq n$, we need a smoothing Wiener filter. The derivation, which is straightforward but somewhat involved, is left for Problem 5.32. The system function of the optimum smoothing filter is

$$H_c^{[D]}(z)=\frac{3}{8}\left(\frac{z^D}{1-\frac{1}{2}z^{-1}}+\frac{2^D\sum_{l=0}^{-D-1}2^l z^{-l}}{1-\frac{1}{2}z^{-1}}-\frac{4}{5}\frac{2^D\sum_{l=0}^{-D-1}2^l z^{-l-1}}{1-\frac{1}{2}z^{-1}}\right) \tag{5.5.54}$$

where $D<0$. To find the impulse response for $D=-2$, we invert (5.5.54). This gives

$$h_c^{[-2]}(k)=\frac{3}{32}\delta(k)+\frac{51}{320}\delta(k-1)+\frac{39}{128}(\frac{1}{2})^{k-2}u(k-2) \tag{5.5.55}$$

and if we express $r_{yx}(k-2)$ in a similar form, we can compute the MMSE

$$P_c^{[-2]}=1-\frac{3}{50}-\frac{51}{400}-(\frac{39}{128})\frac{5}{3}=\frac{39}{128}=0.3047 \tag{5.5.56}$$

which is less than $P_c=0.375$. This should be expected since the smoothing Wiener filter uses more information than the Wiener filter (i.e., when $D=0$). In fact it can be shown that

$$\lim_{D\to-\infty}P_c^{[D]}=P_{nc} \quad \text{and} \quad \lim_{D\to-\infty}h_c^{[D]}(n)=h_{nc}(n) \tag{5.5.57}$$

which is illustrated in Figure 5.20. Figure 5.21 shows the impulse responses of the various optimum IIR filters designed in this example. Interestingly, all are obtained by shifting and truncating the impulse response of the optimum noncausal IIR filter.

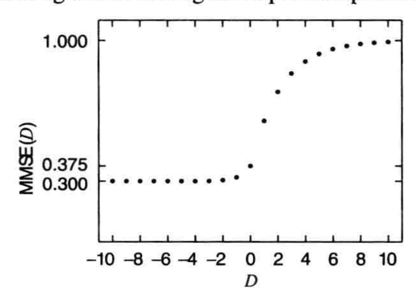


FIGURE 5.20
MMSE as a function of the time shift D.

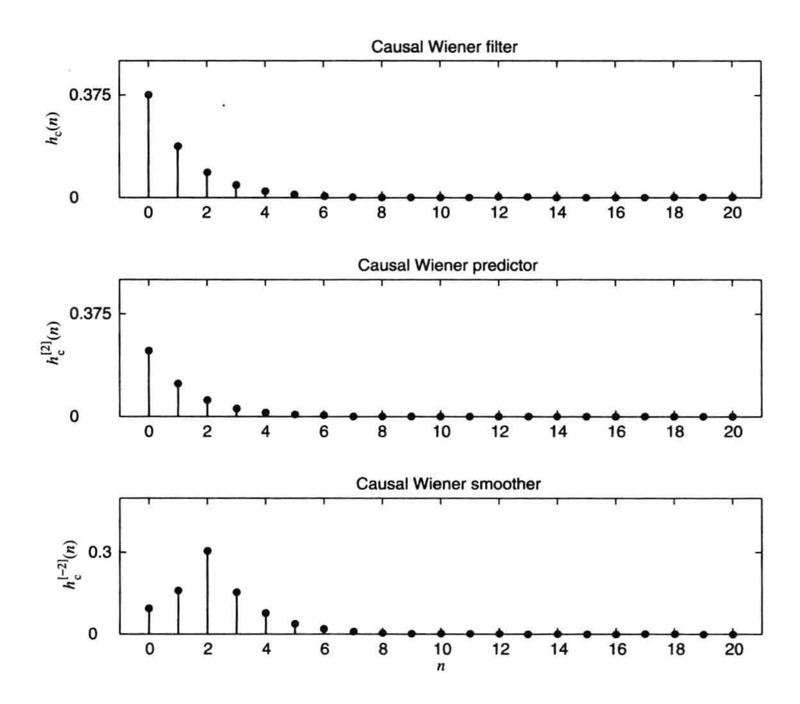


FIGURE 5.21
Impulse response of optimum filters for pure filtering, prediction, and smoothing.

FIR filter. The M th-order FIR filter is obtained by solving the linear system

$$\boldsymbol{Rh} = \boldsymbol{d}$$

where

$$\boldsymbol{R} = \text{Toeplitz}(1+\sigma_v^2,\ \alpha,\ \cdots,\ \alpha^{M-1})$$

and

$$\boldsymbol{d} = [1\ \ \alpha\ \ \cdots\ \ \alpha^{M-1}]^T$$

The MMSE is

$$P_o = r_y(0) - \sum_{k=0}^{M-1} h_o(k) r_{yx}(k)$$

and is shown in Figure 5.22 as a function of the order M together with P_C and P_{nc}. We notice that an optimum FIR filter of order M=4 provides satisfactory performance. This can be explained by noting that the impulse response of the causal optimum IIR filter is negligible for $n > 4$.

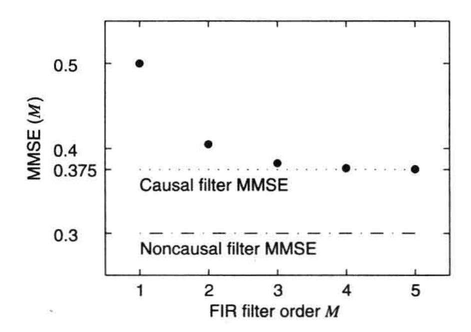


FIGURE 5.22
MMSE as a function of the optimum FIR filter order M.

5.5.4 Linear Prediction Using the Infinite Past—Whitening

The one-step forward IIR linear predictor is a causal IIR optimum filter with desired response $y(n) \triangleq x(n+1)$. The prediction error is

$$e^f(n+1) = x(n+1) - \sum_{k=0}^{\infty} h_{lp}(k) x(n-k) \tag{5.5.58}$$

where

$$H_{lp}(z) = \sum_{k=0}^{\infty} h_{lp}(k) z^{-k} \tag{5.5.59}$$

is the system function of the optimum predictor. Since $y(n)=x(n+1)$, we have $r_{yx}(l) = r_x(l+1)$ and $R_{yx}(z) = zR_x(z)$. Hence, the optimum predictor is

$$H_{lp}(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{z\sigma_x^2 H_x(z) H_x(z^{-1})}{H_x(z^{-1})} \right]_+ = \frac{[zH_x(z)]_+}{H_x(z)} = \frac{zH_x(z) - z}{H_x(z)}$$

and the prediction error filter (PEF) is

$$H_{\text{PEF}}(z) = \frac{E^f(z)}{X(z)} = 1 - z^{-1} H_{lp}(z) = \frac{1}{H_x(z)} \tag{5.5.60}$$

that is, *the one-step IIR linear predictor of a regular process is identical to the whitening filter of the process.* Therefore, the prediction error process is white, and the prediction error filter is minimum-phase. We will see that the efficient solution of optimum filtering problems includes as a prerequisite the solution of a linear prediction problem. Furthermore, algorithms for linear prediction provide a convenient way to perform spectral factorization in practice.

The MMSE is

$$P_o^f = \frac{1}{2\pi j} \oint_C \left\{ R_x(z) - z\left[1 - \frac{1}{H_x(z)}\right] z^{-1} R_x^*\left(\frac{1}{z^*}\right)\right\} z^{-1} dz$$

$$= \frac{1}{2\pi j} \oint_C R_x(z) \frac{1}{H_x(z)} z^{-1} dz \tag{5.5.61}$$

$$= \sigma_x^2 \frac{1}{2\pi j} \oint_C H_x^*\left(\frac{1}{z^*}\right) z^{-1} dz = \sigma_x^2$$

because

$$\frac{1}{2\pi j} \oint_C H_x^*\left(\frac{1}{z^*}\right) z^{-1} dz = h_x(0) = 1$$

From (5.5.61) we have

$$P_o^f = \sigma_x^2 = \exp\left[\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln R_x(e^{j\omega}) d\omega\right] \tag{5.5.62}$$

which is known as the *Kolmogorov-Szegö* formula.

We can easily see that the D-step predictor ($D > 0$) is given by

$$H_D(z) = \frac{[z^D H_x(z)]_+}{H_x(z)} = \frac{1}{H_x(z)} \sum_{k=D}^{\infty} h_x(k) z^{-k+D} \tag{5.5.63}$$

but is not guaranteed to be minimum-phase for $D \neq 1$.

EXAMPLE 5.5.2. Consider a minimum-phase AR(2) process

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \omega(n)$$

where $\omega(n) \sim \text{WN}(0, \sigma_\omega^2)$. The complex PSD of the process is

$$R_x(z) = \frac{\sigma_x^2}{A(z)A(z^{-1})} \triangleq \sigma_x^2 H_x(z) H_x(z^{-1})$$

where $A(z) \triangleq 1 - a_1 z^{-1} - a_2 z^{-2}$ and $\sigma_x^2 = \sigma_\omega^2$. The one-step forward predictor is given by

$$H_{\text{lp}}(z) = z - \frac{z}{H_x(z)} = z - zA(z) = a_1 + a_2 z^{-1}$$

or

$$\hat{x}(n+1) = a_1 x(n) + a_2 x(n-1)$$

as should be expected because the present value of the process depends only on the past two values. Since the excitation $w(n)$ is white and cannot be predicted from the present or previous values of the signal $x(n)$, it is equal to the prediction error $e^f(n)$. Therefore, $\sigma_{e^f}^2 = \sigma_w^2$, as expected from (5.5.62). This shows that the MMSE of the one-step linear predictor depends on the SFM of the process $x(n)$. It is maximum for a white noise process, which is clearly unpredictable.

Predictable processes. A random process $x(n)$ is said to be (exactly) *predictable* if $P_e = E\{|e^f(n)|^2\} = 0$. We next show that a process $x(n)$ is predictable if and only if its PSD consists of impulses, that is,

$$R_x(e^{j\omega}) = \sum_k A_k \delta(\omega - \omega_k) \tag{5.5.64}$$

or in other words, $x(n)$ is a harmonic process. For this reason harmonic processes are also known as *deterministic* processes. From (5.5.60) we have

$$P_e = E\{|e^f(n)|^2\} = \int_{-\pi}^{\pi} |H_{\text{PEF}}(e^{j\omega})|^2 R_x(e^{j\omega}) d\omega \tag{5.5.65}$$

where $H_{\text{PEF}}(e^{j\omega})$ is the frequency response of the prediction error filter. Since $R_x(e^{j\omega}) \geq 0$, the integral in (5.5.65) is zero if and only if $|H_{\text{PEF}}(e^{j\omega})|^2 R_x(e^{j\omega}) = 0$. This is possible only if $R_x(e^{j\omega})$ is a linear combination of impulses, as in (5.5.64), and $e^{j\omega_k}$ are the zeros of $H_{\text{PEF}}(z)$ on the unit circle (Papoulis 1985).

From the Wold decomposition theorem (see Section 3.1.3) we know that every random process can be decomposed into two components that are mutually orthogonal: (1) a regular component with continuous PSD that can be modeled as the response of a minimum-phase system to white noise and (2) a predictable process that can be

exactly predicted from a linear combination of past values. This component has a line PSD and is essentially a harmonic process. A complete discussion of this subject can be found in Papoulis (1985, 1991) and Therrien (1992).

5.6 Inverse Filtering and Deconvolution

In many practical applications, a signal of interest passes through a distorting system whose output may be corrupted by additive noise. When the distorting system is linear and time-invariant, the observed signal is the convolution of the desired input with the impulse response of the system. Since in most cases we deal with linear and time-invariant systems, the terms *filtering* and *convolution* are often used interchangeably.

Deconvolution is the process of retrieving the *unknown* input of a *known* system by using its observed output. If the system is also unknown, which is more common in practical applications, we have a problem of *blind deconvolution*. The term *blind deconvolution* was introduced in Stockham et al. (1975) for a method used to restore old records. Other applications include estimation of the vocal tract in speech processing, equalization of communication channels, deconvolution of seismic data for the elimination of multiple reflections, and image restoration.

The basic problem is illustrated in Figure 5.23. The output of the unknown LTI system $G(z)$, which is assumed BIBO stable, is given by

$$x(n) = \sum_{k=-\infty}^{\infty} g(k)\omega(n-k) \tag{5.6.1}$$

where $\omega(n) \sim \text{IID}(0, \sigma_\omega^2)$ is a white noise sequence. Suppose that we observe the output $x(n)$ and that we wish to recover the input signal $\omega(n)$, and possibly the system $G(z)$, using the output signal and some statistical information about the input.

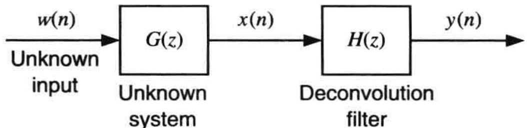


FIGURE 5.23
Basic blind deconvolution model.

If we know the system $G(z)$, the inverse system $H(z)$ is obtained by noticing that perfect retrieval of the input is possible if

$$h(n) * g(n) * w(n) = b_0 w(n-n_0) \tag{5.6.2}$$

where b_0 and n_0 are constants. From (5.6.2), we have $h(n) * g(n) = b_0\delta(n-n_0)$, or equivalently

$$H(z) = b_0 \frac{z^{-n_0}}{G(z)} \tag{5.6.3}$$

which provides the system function of the inverse system. The input can be recovered by convolving the output with the inverse system $H(z)$. Therefore, the terms *inverse filtering* and *deconvolution* are equivalent for LTI systems.

There are three approaches for blind deconvolution:

- Identify the system $G(z)$, design its inverse system $H(z)$, and then compute the input $w(n)$.
- Identify directly the inverse $H(z) = 1/G(z)$ of the system, and then determine the input $w(n)$.
- Estimate directly the input $w(n)$ from the output $x(n)$.

Any of the above approaches requires either directly or indirectly the estimation of both the magnitude response $|G(e^{j\omega})|$ and the phase response $\measuredangle G(e^{j\omega})$ of the unknown system. In practice, the problem becomes more complicated because the output $x(n)$ is usually corrupted by additive noise. If this noise is uncorrelated with the input signal and the required second-order moments are available, we show how to design an optimum inverse filter that provides an optimum estimate of the input in the presence of noise.

We now discuss the design of optimum inverse filters for linearly distorted signals observed in the presence of additive output noise. The typical configuration is shown in Figure 5.24. Ideally, we would like the optimum filter to restore the distorted signal $x(n)$ to its original value $y(n)$. However, the ability of the optimum filter to attain ideal

performance is limited by three factors. First, there is additive noise $v(n)$ at the output of the system. Second, if the physical system $G(z)$ is causal, its output $s(n)$ is delayed with respect to the input, and we may need some delay z^{-D} to improve the performance of the system. When $G(z)$ is a non-minimum-phase system, the inverse system is either noncausal or unstable and should be approximated by a causal and stable filter. Third, the inverse system may be IIR and should be approximated by an FIR filter.

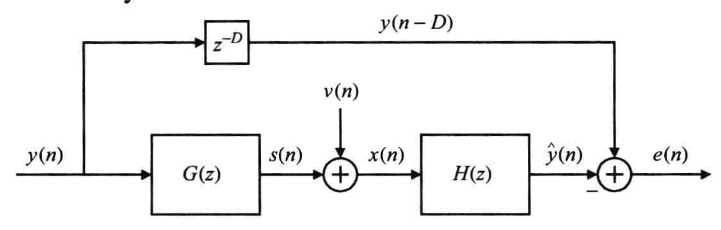


FIGURE 5.24
Typical configuration for optimum inverse system modeling.

The optimum inverse filter is the noncausal Wiener filter

$$H_{\text{nc}}(z) = \frac{z^{-D} R_{yx}(z)}{R_x(z)} \tag{5.6.4}$$

where the term z^{-D} appears because the desired response is $y_D(n) \triangleq y(n-D)$. Since $y(n)$ and $v(n)$ are uncorrelated, we have

$$R_{yx}(z) = R_{ys}(z) \tag{5.6.5}$$

and

$$R_x(z) = G(z)G^*\left(\frac{1}{z^*}\right)R_y(z) + R_v(z) \tag{5.6.6}$$

The cross-correlation between $y(n)$ and $s(n)$

$$R_{ys}(z) = G^*\left(\frac{1}{z^*}\right)R_y(z) \tag{5.6.7}$$

is obtained by using Equation (5.5.18). Therefore, the optimum inverse filter is

$$H_{\text{nc}}(z) = \frac{z^{-D} G^*(1/z^*) R_y(z)}{G(z)G^*(1/z^*)R_y(z) + R_v(z)} \tag{5.6.8}$$

which, in the absence of noise, becomes

$$H_{\text{nc}}(z) = \frac{z^{-D}}{G(z)} \tag{5.6.9}$$

as expected. The behavior of the optimum inverse system is illustrated in the following example.

EXAMPLE 5.6.1 Let the system $G(z)$ be an all-zero non-minimum-phase system given by

$$G(z) = \frac{1}{5}(-3z + 7 - 2z^{-1}) = -\frac{3}{5}(1 - \frac{1}{3}z^{-1})(z-2)$$

Then the inverse system is given by

$$H(z) = G^{-1}(z) = \frac{5}{-3z + 7 - 2z^{-1}} = \frac{1}{1 - \frac{1}{3}z^{-1}} - \frac{1}{1 - 2z^{-1}}$$

which is stable if the ROC is $-1/3 < |z| < 2$. Therefore, the impulse response of the inverse system is

$$h(n) = \begin{cases} \left(\frac{1}{3}\right)^n & n \geq 0 \\ 2^n & n < 0 \end{cases}$$

which is noncausal and stable.

Following the discussion given in this section, we want to design an optimum inverse system given that $G(z)$ is driven by a white noise sequence $y(n)$ and that the additive noise $v(n)$ is white, that is, $R_y(z) = \sigma_y^2$ and $R_v(z) = \sigma_v^2$. From (5.6.8), the optimum noncausal inverse filter is given by

$$H_{nc}(z) = \frac{z^{-D}}{G(z) + [1/G(z^{-1})](\sigma_v^2/\sigma_y^2)}$$

which can be computed by assuming suitable values for variances σ_y^2 and σ_v^2. Note that if $\sigma_v^2 \ll \sigma_y^2$, that is, for very large SNR, we obtain (5.6.9).

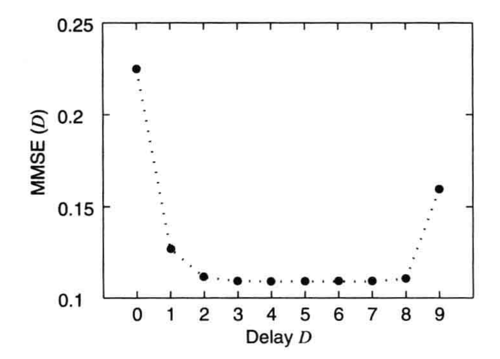


FIGURE 5.25
The inverse filtering MMSE as a function of delay D.

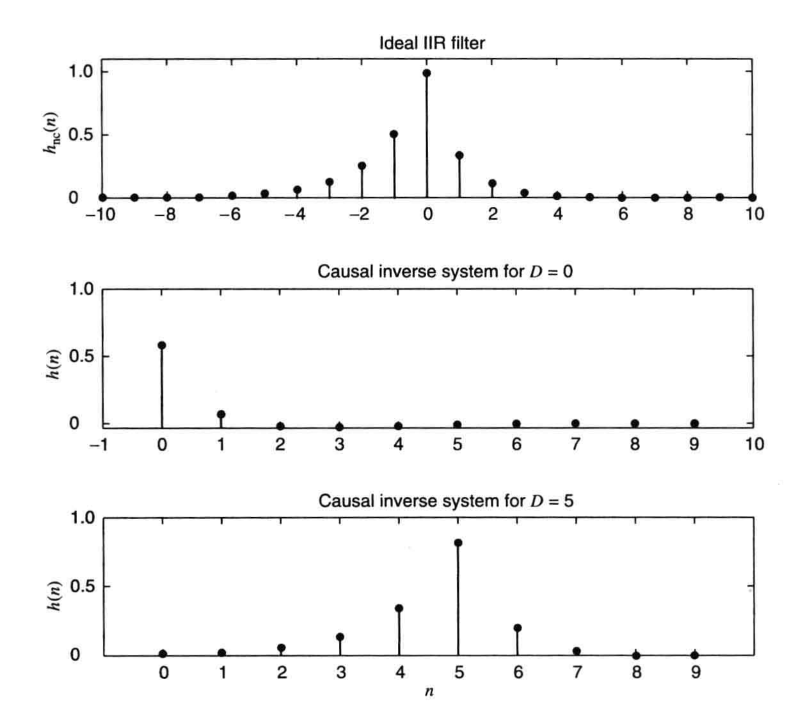


FIGURE 5.26
Impulse responses of optimum inverse filters.

A more interesting case occurs when the optimum inverse filter is FIR, which can be easily implemented. To design this FIR filter, we will need the autocorrelation $r_x(l)$ and the cross-correlation $r_{y_D x}(l)$, where $y_D(n) = y(n - D)$ is the delayed system input sequence. Since

$$R_x(z) = \sigma_y^2 G(z)G(z^{-1}) + \sigma_v^2$$

and

$$R_{y_D x}(l) = \sigma_y^2 z^{-D} G(z^{-1})$$

we have (see Section 2.2.1)

$$r_x(l) = g(l) * g(-l) * r_y(l) + r_v(l) = \sigma_y^2[g(l) * g(-l)] + \sigma_v^2 \delta(l)$$

and

$$r_{y_D x}(l) = g(-l) * r_y(l-D) = \sigma_y^2 g(-l+D)$$

repectively. Now we can determine the optimum FIR filter $\boldsymbol{h}_D$ of length M by constructing an $M \times M$ Toeplitz matrix $\boldsymbol{R}$ from $r_x(l)$ and an $M \times 1$ vector $\boldsymbol{d}$ from $r_{y_D}(l)$ and then solving

$$\boldsymbol{R}\boldsymbol{h}_D = \boldsymbol{d}$$

for various values of D. We can then plot the MMSE as a function of D to determine the best value of D (and the corresponding FIR filter) which will give the smallest MMSE. For example, if $\sigma_y^2 = 1, \sigma_v^2 = 0.1$, and $M = 10$, the correlation functions are

$$r_x(l) = \begin{bmatrix} \frac{6}{25}, & -\frac{7}{5}, & \frac{129}{50}, & -\frac{7}{5}, & \frac{6}{25} \\ & & \uparrow & & \\ & & l=0 & & \end{bmatrix} \text{ and } r_{y_D x}(l) = \begin{bmatrix} -\frac{2}{5}, & \frac{7}{5}, & -\frac{3}{5} \\ & \uparrow & \\ & l=D & \end{bmatrix}$$

The resulting MMSE as a function of D is shown in Figure 5.25, which indicates that the best value of D is approximately $M/2$. Finally, plots of impulse responses of the inverse system are shown in Figure 5.26. The first plot shows the noncausal $h(n)$, the second plot shows the causal FIR system $h_0(n)$ for $D = 0$, and the third plot shows the causal FIR system $h_D(n)$ for $D = 5$. It is clear that the optimum delayed FIR inverse filter for $D \simeq M/2$ closely matches the impulse response of the inverse filter $h(n)$.

5.7 Summary

In this chapter, we discussed the theory and application of optimum linear filters designed by minimizing the MSE criterion of performance. Our goal was to explain the characteristics of each criterion, emphasize when its use made sense, and illustrate its meaning in the context of practical applications.

We started with linear processors that formed an estimate of the desired response by combining a set of different signals (data) and showed that the parameters of the optimum processor can be obtained by solving a linear system of equations (normal equations). The matrix and the right-hand side vector of the normal equations are completely specified by the second-order moments of the input data and the desired response. Next, we used the developed theory to design optimum FIR filters, linear signal estimators, and linear predictors.

We emphasized the case of stationary stochastic processes and showed that the resulting optimum estimators are time-invariant. Therefore, we need to design only one optimum filter that can be used to process all realizations of the underlying stochastic processes. Although another filter may perform better for some realizations, that is, the estimated MSE is smaller than the MMSE, on average (i.e., when we consider all possible realizations), the optimum filter is the best.

We showed that the performance of optimum linear filters improves as we increase the number of filter coefficients. Therefore, the noncausal IIR filter provides the best possible performance and can be used as a yardstick to assess other filters. Because IIR filters involve an infinite number of parameters, their design involves linear equations with an infinite number of unknowns. For stationary processes, these equations take the form of a convolution equation that can be solved using z-transform techniques. If we use a pole-zero structure, the normal equations become nonlinear and the design of the optimum filter is complicated by the presence of multiple local minima.

Then we discussed the design of optimum filters for inverse system modeling and blind deconvolution, and we provided a detailed discussion of their use in the important practical application of channel equalization for data

transmission systems.

Problems

5.1 Let $\boldsymbol{x}$ be a random vector with mean $E\{\boldsymbol{x}\}$. Show that the linear MMSE estimate $\hat{y}$ of a random variable y using the data vector $\boldsymbol{x}$ is given by $\hat{y} = y_o + \boldsymbol{c}^{\mathrm{H}}\boldsymbol{x}$, where $y_o = E\{y\} - \boldsymbol{c}^{\mathrm{H}}E\{\boldsymbol{x}\}$, $\boldsymbol{c} = \boldsymbol{R}^{-1}\boldsymbol{d}$, $\boldsymbol{R} = E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\}$, and $\boldsymbol{d} = E\{\boldsymbol{x}y^*\}$.

5.2 Consider an optimum FIR filter specified by the input correlation matrix $\boldsymbol{R} =$ Toeplitz $\{1,\ 1/4\}$ and cross-correlation vector $\boldsymbol{d} = [1\ \ 1/2]^{\mathrm{T}}$.

(*a*) Determine the optimum impulse response $\mathbf{c}_o$ and the MMSE P_o.

(*b*) Express $\mathbf{c}_o$ and P_o in terms of the eigenvalues and eigenvectors of $\boldsymbol{R}$.

5.3 Repeat Problem 5.2 for a third-order optimum FIR filter.

5.4 A process $y(n)$ with the autocorrelation $r_y(l) = a^{|l|}, -1 < a < 1$, is corrupted by additive, uncorrelated white noise $v(n)$ with variance σ_v^2. To reduce the noise in the observed process $x(n) = y(n) + v(n)$, we use a first-order Wiener filter.

(*a*) Express the coefficients $c_{o,1}$ and $c_{o,2}$ and the MMSE P_o in terms of parameters a and σ_v^2.

(*b*) Compute and plot the PSD of $x(n)$ and the magnitude response $|C_o(e^{j\omega})|$ of the filter when $\sigma_v^2 = 2$, for both $a = 0.8$ and $a = -0.8$, and compare the results.

(*c*) Compute and plot the processing gain of the filter for $a = -0.9,\ -0.8,\ -0.7,\ \cdots,\ 0.9$ as a function of a and comment on the results.

5.5 Consider the harmonic process $y(n)$ and its noise observation $x(n)$ given in Example 5.3.1.

(*a*) Show that $r_y(l) = 1/2\, A^2 \cos \omega_0 l$.

(*b*) Write a Matlab function `h = opt_fir(A, f0, var_v,M)` to design an M th-order optimum FIR filter impulse response $h(n)$. Use the `toeplitz` function from MATLAB to generate correlation matrix $\boldsymbol{R}$.

(*c*) Determine the impulse response of a 20th-order optimum FIR filter for $A = 0.5$, $f_0 = 0.05$, and $\sigma_v^2 = 0.5$.

(*d*) Using MATLAB, determine and plot the magnitude response of the above-designed filter, and verify your results with those given in Example 5.3.1.

5.6 Consider a "desired" signal $s(n)$ generated by the process $s(n) = -0.8w(n-1) + w(n)$, where $w(n) \sim \mathrm{WN}(0, \sigma_w^2)$. This signal is passed through the causal system $H(z) = 1 - 0.9z^{-1}$ whose output $y(n)$ is corrupted by additive white noise $v(n) \sim WN(0, \sigma_v^2)$. The processes $w(n)$ and $v(n)$ are uncorrelated with $\sigma_w^2 = 0.3$ and $\sigma_v^2 = 0.1$.

(*a*) Design a second-order optimum FIR filter that estimates $s(n)$ from the signal $x(n) = y(n) + v(n)$ and determine $\mathbf{c}_o$ and P_o.

(*b*) Plot the error performance surface, and verify that it is quadratic and that the optimum filter points to its minimum.

(*c*) Repeat part (*a*) for a third-order filter, and see whether there is any improvement.

5.7 Repeat Problem 5.6, assuming that the desired signal is generated by $s(n) = -0.8s(n-1) + w(n)$.

5.8 Repeat Problem 5.6, assuming that $H(z) = 1$.

5.9 A stationary process $x(n)$ is generated by the difference equation $x(n) = \rho x(n-1) + w(n)$, where $w(n) \sim \mathrm{WN}(0, \sigma_w^2)$.

(*a*) Show that the correlation matrix of $x(n)$ is given by

$$\boldsymbol{R}_x = \frac{\sigma_w^2}{1-\rho^2}\text{Toeplitz}\{1,\ \rho,\ \rho^2,\ \ldots,\ \rho^{M-1}\}$$

(*b*) Show that the M th-order FLP is given by $a_1^{(M)} = -\rho$, $a_k^{(M)} = 0$ for $k > 1$ and the MMSE is $P_M^f = \sigma_w^2$.

5.10 Using Parseval's theorem, show that (5.3.18) can be written as (5.4.21) in the frequency domain.

5.11 By differentiating (5.3.21) with respect to $H(e^{j\omega})$, derive the frequency response function $H_o(e^{j\omega})$ of the optimum filter in terms of $R_{yx}(e^{j\omega})$ and $R_x(e^{j\omega})$.

5.12 A conjugate symmetric linear smoother is obtained from (5.4.12) when $M = 2L$ and $i = L$. If the process $x(n)$ is stationary, then, using $\bar{\boldsymbol{R}}\boldsymbol{J} = \boldsymbol{J}\bar{\boldsymbol{R}}^*$, show that $\bar{\boldsymbol{c}} = \boldsymbol{J}\bar{\boldsymbol{c}}^*$.

5.13 Let $\bar{\boldsymbol{Q}}$ and $\bar{\Lambda}$ be the matrices from the eigendecomposition of $\bar{\boldsymbol{R}}$, that is, $\bar{\boldsymbol{R}} = \bar{\boldsymbol{Q}}\bar{\Lambda}\bar{\boldsymbol{Q}}^{\mathrm{H}}$.

(*a*) Substitute $\boldsymbol{R}$ into (5.4.20) and (5.4.27) to prove (5.4.43) and (5.4.44).

(*b*) Generalize the above result for a jth-order linear signal estimator $\boldsymbol{c}^{(j)}(n)$; that is, prove that

$$\boldsymbol{c}^{(j)}(n) = P_o^{(j)}(n)\sum_{i=1}^{M+1}\frac{1}{\bar{\lambda}_i}\bar{\boldsymbol{q}}_i\bar{q}_{i,j}$$

5.14 Let $\tilde{\boldsymbol{R}}(n)$ be the inverse of the correlation matrix $\bar{\boldsymbol{R}}(n)$ given in (5.4.11).
(*a*) Using (5.4.12), show that the diagonal elements of $\tilde{\boldsymbol{R}}(n)$ are given by

$$<\tilde{\boldsymbol{R}}(n)>_{i,i} = \frac{1}{P^{(i)}(n)} \qquad 1 \le i \le M+1$$

(*b*) Furthermore, show that

$$\boldsymbol{c}^{(i)}(n) = \frac{\tilde{\boldsymbol{r}}_i(n)}{<\tilde{\boldsymbol{R}}(n)>_{i,i}} \qquad 1 \le i \le M+1$$

where $\tilde{\boldsymbol{R}}_i(n)$ is the *i*-th column of $\tilde{\boldsymbol{R}}(n)$.

5.15 The first five samples of the autocorrelation sequence of a signal $x(n)$ are $r(0)=1$, $r(1)=0.8$, $r(2)=0.6$, $r(3)=0.4$, and $r(4)=0.3$. Compute the FLP, the BLP, the optimum symmetric smoother, and the corresponding MMSE (*a*) by using the normal equations method and (*b*) by using the inverse of the normal equations matrix.

5.16 For the symmetric, Toeplitz autocorrelation matrix $\boldsymbol{R}=$ Toeplitz $\{r(0),\ r(1),\ r(2)\} = r(0)\times$ Toeplitz $\{1,\ \rho_1,\ \rho_2\}$ with $\boldsymbol{R}=\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{H}}$ and $\boldsymbol{D}=\mathrm{diag}\{\xi_1,\ \xi_2,\ \xi_3\}$, the following conditions are equivalent:

- $\boldsymbol{R}$ is positive definite.
- $\xi_i > 0$ for $1 \le i \le 3$.
- $|k_i| < 1$ for $1 \le i \le 3$.

Determine the values of ρ_1 and ρ_2 for which $\boldsymbol{R}$ is positive definite, and plot the corresponding area in the $(\rho_1,\ \rho_2)$ plane.

5.17 Prove the first equation in (5.4.45) by rearranging the FLP normal equations in terms of the unknowns $P_o^{\mathrm{f}}(n),\ a_1(n),\ \cdots,\ a_M(n)$ and then solve for $P_o^{\mathrm{f}}(n)$, using Cramer's rule. Repeat the procedure for the second equation.

5.18 Consider the signal $x(n)=y(n)+v(n)$, where $y(n)$ is a useful random signal corrupted by noise $v(n)$. The processes $y(n)$ and $v(n)$ are uncorrelated with PSDs

$$R_y(e^{j\omega}) = \begin{cases} 1 & 0 \le |\omega| \le \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| \le \pi \end{cases}$$

and

$$R_v(e^{j\omega}) = \begin{cases} 1 & \frac{\pi}{4} \le |\omega| \le \frac{\pi}{2} \\ 0 & 0 \le |\omega| < \frac{\pi}{4} \text{ and } \frac{\pi}{2} < |\omega| \le \pi \end{cases}$$

respectively. (*a*) Determine the optimum IIR filter and find the MMSE. (*b*) Determine a third order optimum FIR filter and the corresponding MMSE. (*c*) Determine the noncausal optimum FIR filter defined by

$$\hat{y}(n) = h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1)$$

5.19 Consider the ARMA (1,1) process $x(n)=0.8x(n-1)+w(n)+0.5w(n-1)$, where $w(n)\sim WGN(0,1)$. (*a*) Determine the coefficients and the MMSE of (1) the one-step ahead FLP $\hat{x}(n)=a_1x(n-1)+a_2x(n-2)$ and (2) the two-step ahead FLP $\hat{x}(n+1)=a_1x(n-1)+a_2x(n-2)$. (*b*) Check if the obtained prediction error filters are minimum-phase, and explain your findings.

5.20 Consider a random signal $x(n)=s(n)+v(n)$, where $v(n)\sim WGN(0,1)$ and $s(n)$ is the AR(1) process $s(n)=0.9s(n-1)+w(n)$, where $w(n)\sim WGN(0,\ 0.64)$. The signals $s(n)$ and $v(n)$ are uncorrelated. (*a*) Determine and plot the autocorrelation $r_s(l)$ and the PSD $R_s(e^{j\omega})$ of $s(n)$. (*b*) Design a second-order optimum FIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE? (*c*) Design an optimum IIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE?

5.21 A useful signal $s(n)$ with PSD $R_s(z)=[(1-0.9z^{-1})(1-0.9z)]^{-1}$ is corrupted by additive uncorrelated noise $v(n)\sim WN(0,\sigma^2)$. (*a*) The resulting signal $x(n)=s(n)+v(n)$ is passed through a causal filter with system function $H(z)=(1-0.8z^{-1})^{-1}$. Determine (1) the SNR at the input, (2) the SNR at the output, and (3) the processing gain, that is, the improvement in SNR. (*b*) Determine the causal optimum filter and compare its performance with that of the filter in (*a*).

5.22 A useful signal $s(n)$ with PSD $R_s(z)=0.36[(1-0.8z^{-1})(1-0.8z)]^{-1}$ is corrupted by additive uncorrelated noise $v(n)\sim WN(0,1)$. Determine the optimum noncausal and causal IIR filters, and compare their performance by examining the MMSE and their magnitude response. *Hint:* Plot the magnitude responses on the same graph with the PSDs of signal and noise.

5.23 Consider a process with PSD $R_x(z)=\sigma^2 H_x(z)H_x(z^{-1})$. Determine the *D*-step ahead linear predictor, and show that the MMSE

is given by $P^{(D)} = \sigma^2 \sum_{n=0}^{D-1} h_x^2(n)$. Check your results by using the PSD $R_x(z) = (1-a^2)[(1-az^{-1})(1-az)]^{-1}$.

5.24 Let $x(n) = s(n) + v(n)$ with $R_v(z) = 1$, $R_{sv}(z) = 0$, and

$$R_s(z) = \frac{0.75}{(1-0.5z^{-1})(1-0.5z)}$$

Determine the optimum filters for the estimation of $s(n)$ and $s(n-2)$ from $\{x(k)\}_{-\infty}^{n}$ and the corresponding MMSEs.

5.25 For the random signal with PSD

$$R_x(z) = \frac{(1-0.2z^{-1})(1-0.2z)}{(1-0.9z^{-1})(1-0.9z)}$$

determine the optimum two-step ahead linear predictor and the corresponding MMSE.

5.26 Repeat Problem5.25 for

$$R_x(z) = \frac{1}{(1-0.2z^{-1})(1-0.2z)(1-0.9z^{-1})(1-0.9z)}$$

5.27 Let $x(n) = s(n) + v(n)$ with $v(n) \sim WN(0,1)$ and $s(n) = 0.6s(n-1) + w(n)$, where $w(n) \sim WN(0, 0.82)$. The processes $s(n)$ and $v(n)$ are uncorrelated. Determine the optimum filters for the estimation of $s(n)$, $s(n+2)$, and $s(n-2)$ from $\{x(k)\}_{-\infty}^{n}$ and the corresponding MMSEs.

5.28 Repeat Problem 5.27 for $R_s(z) = [(1-0.5z^{-1})(1-0.5z)]^{-1}$, $R_v(z) = 5$, and $R_{sv}(z) = 0$.

5.29 Consider the random sequence $x(n)$ generated in Example 5.4.2

$$x(n) = w(n) + \frac{1}{2} w(n-1)$$

where $w(n)$ is $WN(0,1)$. Generate K sample functions $\{w_k(n)\}_{n=0}^{N}$, $k = 1, \cdots, K$ of $\omega(n)$, in order to generate K sample functions $\{x_k(n)\}_{n=0}^{N}$, $k = 1, \cdots, K$ of $x(n)$.

(*a*) Use the second-order FLP a_k to obtain predictions $\{\hat{x}_k^{\mathrm{f}}(n)\}_{n=2}^{N}$ of $x_k(n)$, for $k = 1, \cdots, K$. Then determine the average error

$$\hat{P}^{\mathrm{f}} = \frac{1}{N-1} \sum_{n=2}^{N} |x_k(n) - \hat{x}_k^{\mathrm{f}}(n)|^2 \qquad k = 1, \cdots, K$$

and plot it as a function of *k*. Compare it with P_o^{f}.

(*b*) Use the second-order BLP b_k to obtain predictions $\{\hat{x}_k^{\mathrm{b}}(n)\}_{n=0}^{N-2}$, $k = 1, \cdots, K$ of $x_k(n)$. Then determine the average error

$$\hat{P}^{\mathrm{b}} = \frac{1}{N-1} \sum_{n=0}^{N-2} |x_k(n) - \hat{x}_k^{b}(n)|^2 \qquad k = 1, \cdots, K$$

and plot it as a function of *k*. Compare it with P_o^{b}.

(*c*) Use the second-order symmetric linear smoother c_k to obtain smooth estimates $\{\hat{x}_k^{c}(n)\}_{n=0}^{N-2}$ of $x_k(n)$ for $k = 1, \cdots, K$. Determine the average error

$$\hat{P}^{\mathrm{s}} = \frac{1}{N-1} \sum_{n=1}^{N-1} |x_k(n) - \hat{x}_k^{c}(n)|^2 \qquad k = 1, \cdots, K$$

and plot it as a function of k. Compare it with P_o^{s}.

5.30 Let $x(n) = y(n) + v(n)$ be a wide-sense stationary process. The linear, symmetric smoothing filter estimator of $y(n)$ is given by

$$\hat{y}(n) = \sum_{k=-L}^{L} h(k)x(n-k)$$

(*a*) Determine the normal equations for the optimum MMSE filter.

(*b*) Show that the smoothing filter $\mathbf{c}_o^{\mathrm{s}}$ has linear phase.

(*c*) Use the Lagrange multiplier method to determine the MMSE *M* th-order estimator $\hat{y}(n) = \mathbf{c}^H \mathbf{x}(n)$, where $M = 2L+1$, when the filter vector **c** is constrained to be conjugate symmetric, that is, $c = Jc^*$. Compare the results with those obtained in part (*a*).

5.31 Consider the causal prediction filter discussed in Example 5.5.1 To determine $H_c^{[D]}(z)$, first compute the causal part of the

z -transform $[R'_{y\omega}(z)]_+$. Next compute $H_c^{[D]}(z)$ by using (5.5.21).

(*a*) Determine $h_c^{[D]}(n)$.

(*b*) Using the above $h_c^{[D]}(n)$, show that

$$P_c^{[D]} = 1 - \frac{5}{8}\left(\frac{4}{5}\right)^{2D}$$

5.32 Consider the causal smoothing filter discussed in Example 5.5.1.

(*a*) Using $[r'_{y\omega}(l)]_+ = r_{y\omega}(l+D)u(l)$, $D<0$, show that $[r'_{y\omega}(l)]_+$ can be put in the form

$$[r'_{y\omega}(l)]_+ = \frac{3}{5}\left(\frac{4}{5}\right)^{l+D}u(l+D) + \frac{3}{5}(2^{l+D})[u(l)-u(l+D)] \qquad D<0$$

(*b*) Hence, show that $[R'_{y\omega}(z)]_+$ is given by

$$[R'_{y\omega}(z)]_+ = \frac{3}{5}\frac{z^D}{1-\frac{4}{5}z^{-1}} + \frac{3}{5}(2^D)\sum_{l=0}^{-D-1} 2^l z^{-l}$$

(*c*) Finally using (5.5.21), prove (5.5.54).

5.33 In this problem, we will prove (5.5.57)

(*a*) Starting with (5.5.42), show that $[R'_{y\omega}(z)]_+$ can also be put in the form

$$[R'_{y\omega}(z)]_+ = \frac{3}{5}\left(\frac{z^D}{1-\frac{4}{5}z^{-1}} + \frac{2^D - z^D}{1-2z^{-1}}\right)$$

(*b*) Now, using (5.5.21), show that

$$H_c^{[D]}(z) = \frac{3}{8}\left[\frac{2^D(1-\frac{4}{5}z^{-1}) + \frac{3}{5}z^{D-1}}{(1-\frac{4}{5}z^{-1})(1-2z^{-1})}\right]$$

hence, show that

$$\lim_{D\to-\infty} H_c^{[D]}(z) = \frac{9}{40}\left[\frac{z^D}{(1-\frac{4}{5}z^{-1})(1-2z^{-1})}\right] = z^D H_{nc}(z)$$

(c) Finally, show that $\lim_{D\to\infty} P_c^{[D]} = P_{nc}$.

5.34 Consider the block diagram of a simple communication system shown in Figure 5.27. The information resides in the signal $s(n)$ produced by exciting the system $H_1(z) = 1/(1+0.95z^{-1})$ with the process $\omega(n) \sim WGN(0,\ 0.3)$. The signal $s(n)$ propagates through the channel $H_2(z) = 1/(1-0.85z^{-1})$, and is corrupted by the additive noise process $v(n) \sim WGN(0, 0.1)$, which is uncorrelated with $\omega(n)$. (*a*) Determine a second-order optimum FIR filter $(M=2)$ that estimates the signal $s(n)$ from the received signal $x(n) = z(n) + v(n)$. What is the corresponding MMSE P_o? (*b*) Plot the error performance surface and verify that the optimum filter corresponds to the bottom of the "bowl." (*c*) Use a Monte Carlo simulation (100 realizations with a 1000-sample length each) to verify the theoretically obtained MMSE in part (*a*). (*d*) Repeat part (*a*) for $M=3$ and check if there is any improvement. *Hint:* To compute the autocorrelation of $z(n)$, notice that the output of $H_1(z)H_2(z)$ is an AR(2) process.

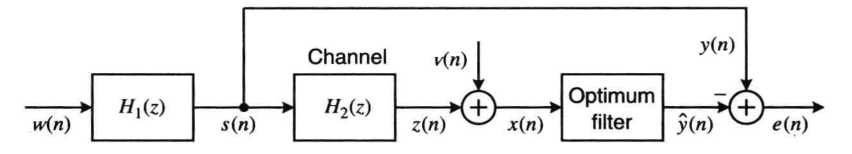


FIGURE 5.38
Block diagram of simple communication system used in Problem 5.34.

5.35 Determine the matched filter for the deterministic pulse $s(n) = \cos \omega_0 n$ for $0 \le n \le M-1$ and zero elsewhere when the noise is (a) white with variance σ_v^2 and (b) colored with autocorrelation $r_v(l) = \sigma_v^2 \rho^{|l|}/(1-\rho^2)$, $-1 < \rho < 1$. Plot the frequency response of the filter and superimpose it on the noise PSD, for $\omega_0 = \pi/6$, $M = 12$, $\sigma_v^2 = 1$, and $\rho = 0.9$. Explain the shape of the obtained response. (c) Study the effect of the SNR in part (a) by varying the value of σ_v^2. (d) Study the effect of the noise correlation in part (c) by varying the value of ρ.

5.36 In this problem we formulate the design of optimum linear signal estimators (LSE) using a constrained optimization framework. To this end we consider the estimator $e(n) = c_0^* x(n) + \cdots + c_M^* x(n-M) \triangleq \mathbf{c}^H \mathbf{x}(n)$ and we wish to minimize the output power $E\{|e(n)|^2\} = \mathbf{c}^H \mathbf{R} \mathbf{c}$. To prevent the trivial solution $\mathbf{c} = 0$ we need to impose some constraint on the filter coefficients and use Lagrange multipliers to determine the minimum. Let $\mathbf{u}_i$ be an $M \times 1$ vector with one at the *i*th position and zeros elsewhere. (a) Show that minimizing $\mathbf{c}^H \mathbf{R} \mathbf{c}$ under the linear constraint $\mathbf{u}_i^T \mathbf{c} = 1$ provides the following estimators: FLP if $i = 0$, BLP if $i = M$, and linear smoother if $i \ne 0, M$. (b) Determine the appropriate set of constraints for the *L*-steps ahead linear predictor, defined by $c_0 = 1$ and $\{c_k = 0\}_1^{L-1}$, and solve the corresponding constrained optimization problem. Verify your answer by obtaining the normal equations using the orthogonality principle. (c) Determine the optimum linear estimator by minimizing $\mathbf{c}^H \mathbf{R} \mathbf{c}$ under the quadratic constraints $\mathbf{c}^H \mathbf{c} = 1$ and $\mathbf{c}^H \mathbf{W} \mathbf{c} = 1$ ($\mathbf{W}$ is a positive definite matrix) which impose a constraint on the length of the filter vector.

CHAPTER 6

Algorthms and Structures for Optimum Linear Filters

The design and application of optimum filters involves (1) the solution of the normal equations to determine the optimum set of coefficients, (2) the evaluation of the cost function to determine whether the obtained parameters satisfy the design requirements, and (3) the implementation of the optimum filter, that is, the computation of its output that provides the estimate of the desired response.

The normal equations can be solved by using any general-purpose routine for linear simultaneous equations. However, there are several important reasons to study the normal equations in greater detail in order to develop efficient, special-purpose algorithms for their solution. First, the throughput of several real-time applications can only be served with serial or parallel algorithms that are obtained by exploiting the special structure (e.g., Toeplitz) of the correlation matrix. Second, sometimes we can develop order-recursive algorithms that help us to choose the correct filter order or to stop the algorithm before the manifestation of numerical problems. Third, some algorithms lead to intermediate sets of parameters that have physical meaning, provide easy tests for important properties (e.g., minimum phase), or are useful in special applications (e.g., data compression). Finally, sometimes there is a link between the algorithm for the solution of the normal equations and the structure for the implementation of the optimum filter.

In this chapter, we present different algorithms for the solution of the normal equations, the computation of the minimum mean square error (MMSE), and the implementation of the optimum filter. We start in Section 6.1 with a discussion of some results from matrix algebra that are useful for the development of order-recursive algorithms and introduce an algorithm for the order-recursive computation of the LDL^H decomposition, the MMSE, and the optimum estimate in the general case.

The only assumption we have made so far is that we know the required second-order statistics; hence, the results apply to any linear estimation problem: array processing, filtering, and prediction of nonstationary or stationary processes. In the sequel, we impose additional constraints on the input data vector and show how to exploit them in order to simplify the general algorithms and structures or specify new ones. In Section 6.3, we explore the shift invariance of the input data vector to develop a time-varying lattice-ladder structure for the optimum filter. However, to derive an order-recursive algorithm for the computation of either the direct or lattice-ladder structure parameters of the optimum time-varying filter, we need an analytical description of the changing second-order statistics of the nonstationary input process. Recall that in the simplest case of stationary processes, the correlation matrix is constant and Toeplitz. As a result, the optimum FIR filters and predictors are time-invariant, and their direct or lattice-ladder structure parameters can be computed (only once) using efficient, order-recursive algorithms due to Levinson and Durbin (Section 6.4). Section 6.5 provides a derivation of the lattice-ladder structures for optimum filtering and prediction, their structural and statistical properties, and algorithms for transformations between the various sets of parameters.

6.1 Fundamentals of Order-Recursive Algorithms

The optimum estimate is computed as a sum of products using a linear combiner supplied with the optimum coefficients and the input data. The key characteristic of this approach is that the order of the estimator should be fixed initially, and in case we choose a different order, we have to repeat *all* the computations. Such computational methods are known as *fixed-order algorithms*.

When the order of the estimator becomes a design variable, we need to modify our notation to take this into account. For example, the m th-order estimator $\boldsymbol{c}_m(n)$ is obtained by minimizing $E\{|e_m(n)|^2\}$, where

$$e_m(n) \triangleq y(n) - \hat{y}_m(n) \tag{6.1.1}$$

$$\hat{y}_m(n) \triangleq \boldsymbol{c}_m^H(n)\boldsymbol{x}_m(n) \tag{6.1.2}$$

$$\boldsymbol{c}_m(n) \triangleq [c_1(n)\ c_2^{(m)}(n)\ \cdots\ c(n)]^{\mathrm{T}} \tag{6.1.3}$$

$$\boldsymbol{x}_m(n) \triangleq [x_1(n)\ x_2(n)\ \cdots\ x_m(n)]^{\mathrm{T}} \tag{6.1.4}$$

In general, we use the subscript m to denote the order of a matrix or vector and the superscript m to emphasize that a scalar is a component of an $m \times 1$ vector. We note that these quantities are functions of time n, but sometimes we do not explicitly show this dependence for the sake of simplicity.

If the m th-order estimator $\boldsymbol{c}_m(n)$ has been computed by solving the normal equations, it seems to be a waste of computational power to start from scratch to compute the $(m+1)$ st-order estimator $\boldsymbol{c}_{m+1}(n)$. Thus, we would like to arrange the computations so that the results for order m, that is, $\mathbf{c}_m(n)$ or $\hat{y}_m(n)$, can be used to compute the estimates for order $m+1$, that is, $\boldsymbol{c}_{m+1}(n)$ or $\hat{y}_{m+1}(n)$. The resulting procedures are called *order-recursive algorithms* or *order-updating relations*. Similarly, procedures that compute $\boldsymbol{c}_m(n+1)$ from $\boldsymbol{c}_m(n)$ or $\hat{y}_m(n+1)$ from $\hat{y}_m(n)$ are called *time-recursive algorithms* or *time-updating relations*. Combined order and time updates are also possible. All these updates play a central role in the design and implementation of many optimum and adaptive filters.

In this section, we derive order-recursive algorithms for the computation of the LDL^H decomposition, the MMSE, and the MMSE optimal estimate. We also show that there is no order-recursive algorithm for the computation of the estimator parameters.

6.1.1 Matrix Partitioning and Optimum Nesting

We start by introducing some notation that is useful for the discussion of order-recursive algorithms.[1] Notice that if the order of the estimator increases from m to $m+1$, then the input data vector is augmented with one additional observation x_{m+1}. We use the notation $\boldsymbol{x}_{m+1}^{\lfloor m \rfloor}$ to denote the vector that consists of the first m components and $\boldsymbol{x}_{m+1}^{\lceil m \rceil}$ for the last m components of vector $\mathbf{x}_{m+1}$. The same notation can be generalized to matrices. The $m \times m$ matrix $\boldsymbol{R}_{m+1}^{\lceil m \rceil}$, obtained by the intersection of the first m rows and columns of $\boldsymbol{R}_{m+1}$, is known as the mth-order *leading principal submatrix* of $\boldsymbol{R}_{m+1}$. In other words, if r_{ij} are the elements of $\boldsymbol{R}_{m+1}$, then the elements of $\boldsymbol{R}_{m+1}^{\lceil m \rceil}$ are $r_{ij}, 1 \le i,\ j \le m$. Similarly, $\boldsymbol{R}_{m+1}^{\lfloor m \rfloor}$ denotes the matrix obtained by the intersection of the last m rows and columns of $\boldsymbol{R}_{m+1}$. For example, if $m = 3$ we obtain

$$\boldsymbol{R}_4 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} \tag{6.1.5}$$

(upper left 3×3 block: $\mathbf{R}_4^{\lceil 3 \rceil}$; lower right 3×3 block: $\mathbf{R}_4^{\lfloor 3 \rfloor}$)

which illustrates the *upper left corner* and *lower right corner* partitionings of matrix $\boldsymbol{R}_4$.

Since $\boldsymbol{x}_{m+1}^{\lceil m \rceil} = \boldsymbol{x}_m$, we can easily see that the correlation matrix can be partitioned as

$$\boldsymbol{R}_{m+1} = E\left\{\begin{bmatrix} \boldsymbol{x}_m \\ x_{m+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_m^H & x_{m+1}^* \end{bmatrix}\right\} = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{r}_m^b \\ \boldsymbol{r}_m^{bH} & \rho_m^b \end{bmatrix} \tag{6.1.6}$$

where

$$\mathrm{r}_m^{\mathrm{b}} \triangleq E\{\boldsymbol{x}_m x_{m+1}^*\} \tag{6.1.7}$$

---

[1]All quantities in Sections 6.1 and 6.2 are functions of the time index n. However, for notational simplicity we do not explicitly show this dependence.

and $$\rho_m^b \triangleq E\{|x_{m+1}|^2\} \tag{6.1.8}$$

The result $$\boldsymbol{x}_{m+1}^{[m]} = \boldsymbol{x}_m \Rightarrow \boldsymbol{R}_m = \boldsymbol{R}_{m+1}^{[m]} \tag{6.1.9}$$

is known as the *optimum nesting property* and is instrumental in the development of order- recursive algorithms. Similarly, we can show that $\mathbf{x}_{m+1}^{[m]} = \boldsymbol{x}_m$ implies

$$\boldsymbol{d}_{m+1} = E\{\boldsymbol{x}_{m+1}y^*\} = E\left\{\begin{bmatrix} \boldsymbol{x}_m \\ x_{m+1} \end{bmatrix} y^*\right\} = \begin{bmatrix} \boldsymbol{d}_m \\ d_{m+1} \end{bmatrix} \tag{6.1.10}$$

or $$\mathbf{x}_{m+1}^{[m]} = \boldsymbol{x}_m \Rightarrow \boldsymbol{d}_m = \boldsymbol{d}_{m+1}^{[m]} \tag{6.1.11}$$

that is, the right-hand side of the normal equations also has the optimum nesting property.

Since (6.1.9) and (6.1.11) hold for all $1 \le m \le M$, the correlation matrix $\boldsymbol{R}_M$ and the cross-correlation vector $\boldsymbol{d}_M$ contain the information for the computation of all the optimum estimators $\boldsymbol{c}_m$ for $1 \le m \le M$.

6.1.2 Inversion of Partitioned Hermitian Matrices

Suppose now that we know the inverse $\boldsymbol{R}_m^{-1}$ of the leading principal submatrix $\boldsymbol{R}_{m+1}^{[m]} = \boldsymbol{R}_m$ of matrix $\boldsymbol{R}_{m+1}$ and we wish to use it to compute $\boldsymbol{R}_{m+1}^{-1}$ without having to repeat all the work. Since the inverse $\boldsymbol{Q}_{m+1}$ of the Hermitian matrix $\boldsymbol{R}_{m+1}$ is also Hermitian, it can be partitioned as

$$\boldsymbol{Q}_{m+1} = \begin{bmatrix} \boldsymbol{Q}_m & \boldsymbol{q}_m \\ \boldsymbol{q}_m^H & q_m \end{bmatrix} \tag{6.1.12}$$

Using (6.1.6), we obtain

$$\boldsymbol{R}_{m+1}\boldsymbol{Q}_{m+1} = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{r}_m^b \\ \boldsymbol{r}_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_m & \boldsymbol{q}_m \\ \boldsymbol{q}_m^H & q_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{0}_m \\ \boldsymbol{0}_m^H & 1 \end{bmatrix} \tag{6.1.13}$$

After performing the matrix multiplication, we get

$$\boldsymbol{R}_m\boldsymbol{Q}_m + \boldsymbol{r}_m^b\boldsymbol{q}_m^H = \boldsymbol{I}_m \tag{6.1.14}$$

$$\boldsymbol{r}_m^{bH}\boldsymbol{Q}_m + \rho_m^b\boldsymbol{q}_m^H = \boldsymbol{0}_m^H \tag{6.1.15}$$

$$\boldsymbol{R}_m\boldsymbol{q}_m + \boldsymbol{r}_m^b q_m = \boldsymbol{0}_m \tag{6.1.16}$$

$$\boldsymbol{r}_m^{bH}\boldsymbol{q}_m + \rho_m^b q_m = 1 \tag{6.1.17}$$

where $\boldsymbol{0}_m$ is the $m\times 1$ zero vector. If matrix $\boldsymbol{R}_m$ is invertible, we can solve (6.1.16) for $\boldsymbol{q}_m$

$$\boldsymbol{q}_m = -\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^b q_m \tag{6.1.18}$$

and then substitute into (6.1.17) to obtain q_m as

$$q_m = \frac{1}{\rho_m^b - \boldsymbol{r}_m^{bH}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^b} \tag{6.1.19}$$

assuming that the scalar quantity $\rho_m^b - \boldsymbol{r}_m^{bH}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^b \neq 0$. Substituting (6.1.19) into (6.1.18), we obtain

$$\boldsymbol{q}_m = \frac{-\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}}}{\rho_m^{\mathrm{b}} - \boldsymbol{r}_m^{\mathrm{bH}}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}}} \tag{6.1.20}$$

which, in conjunction with (6.1.14), yields

$$\boldsymbol{Q}_m = \boldsymbol{R}_m^{-1} - \boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}}\boldsymbol{q}_m^{\mathrm{H}} = \boldsymbol{R}_m^{-1} + \frac{\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}}(\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}})^{\mathrm{H}}}{\rho_m^{\mathrm{b}} - \boldsymbol{r}_m^{\mathrm{bH}}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}}} \tag{6.1.21}$$

We note that (6.1.19) through (6.1.21) express the parts of the inverse matrix $\boldsymbol{Q}_{m+1}$ in terms of known quantities. For our purposes, we express the above equations in a more convenient form, using the quantities

$$\boldsymbol{b}_m \triangleq [b_0^{(m)}\ b_1^{(m)}\ \cdots\ b_{m-1}^{(m)}]^{\mathrm{T}} \triangleq -\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}} \tag{6.1.22}$$

and

$$\alpha_m^{\mathrm{b}} \triangleq \rho_m^{\mathrm{b}} - \boldsymbol{r}_m^{\mathrm{bH}}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}} = \rho_m^{\mathrm{b}} + \boldsymbol{r}_m^{\mathrm{bH}}\boldsymbol{b}_m \tag{6.1.23}$$

Thus, if matrix $\boldsymbol{R}_m$ is invertible and $\alpha_m^b \neq 0$, combining (6.1.13) with (6.1.19) through (6.1.23), we obtain

$$\boldsymbol{R}_{m+1}^{-1} = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{r}_m^{\mathrm{b}} \\ \boldsymbol{r}_m^{\mathrm{bH}} & \rho_m^{\mathrm{b}} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{R}_m^{-1} & \boldsymbol{0}_m \\ \boldsymbol{0}_m^{\mathrm{H}} & 0 \end{bmatrix} + \frac{1}{\alpha_m^{\mathrm{b}}}\begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{b}_m^{\mathrm{H}} & 1 \end{bmatrix} \tag{6.1.24}$$

which determines $\boldsymbol{R}_{m+1}^{-1}$ from $\boldsymbol{R}_m^{-1}$ by using a simple rank-one modification known as the *matrix inversion by partitioning lemma* (Noble and Daniel 1988).

Another useful expression for α_m^{b} is

$$\alpha_m^{\mathrm{b}} = \frac{\det \boldsymbol{R}_{m+1}}{\det \boldsymbol{R}_m} \tag{6.1.25}$$

which reinforces the importance of the quantity α_m^{b} for the invertibility of matrix $\boldsymbol{R}_{m+1}$ (see Problem 6.1).

EXAMPLE 6.1.1 Given the matrix

$$\boldsymbol{R}_3 = \left[\begin{array}{cc|c} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \hline \frac{1}{3} & \frac{1}{2} & 1 \end{array}\right] = \begin{bmatrix} \boldsymbol{R}_2 & \boldsymbol{r}_2^{\mathrm{b}} \\ \boldsymbol{r}_2^{\mathrm{bH}} & \rho_2^{\mathrm{b}} \end{bmatrix}$$

and the inverse matrix

$$\boldsymbol{R}_2^{-1} = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}^{-1} = \frac{1}{3}\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}$$

compute matrix $\boldsymbol{R}_3^{-1}$, using the matrix inversion by partitioning lemma.

Solution. To determine $\boldsymbol{R}_3^{-1}$ from the order-updating formula (6.1.24), we first compute

$$\boldsymbol{b}_2 = -\boldsymbol{R}_2^{-1}\boldsymbol{r}_2^{\mathrm{b}} = -\frac{1}{3}\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}\begin{bmatrix} \frac{1}{3} \\ \frac{1}{2} \end{bmatrix} = -\frac{1}{9}\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

and

$$\alpha_2^{\mathrm{b}} = \rho_2^{\mathrm{b}} + \boldsymbol{r}_2^{\mathrm{bH}}\boldsymbol{b}_2 = 1 - \frac{1}{9}\begin{bmatrix} \frac{1}{3} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \frac{20}{27}$$

using(6.1.22) and (6.1.23). Then we compute

$$\boldsymbol{R}_3^{-1} = \frac{1}{3}\left[\begin{array}{cc|c} 4 & -2 & 0 \\ -2 & 4 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] + \frac{27}{20}\left[\begin{array}{c} -\frac{1}{9} \\ -\frac{4}{9} \\ \hline 1 \end{array}\right]\left[\begin{array}{cc|c} -\frac{1}{9} & -\frac{4}{9} & 1 \end{array}\right] = 1/20\begin{bmatrix} 27 & -12 & -3 \\ -12 & 32 & -12 \\ -3 & -12 & 27 \end{bmatrix}$$

using (6.1.24). The reader can easily verify the above calculations using MATLAB.

Following a similar approach, we can show (see Problem 6.2) that the inverse of the lower right corner partitioned matrix $\boldsymbol{R}_{m+1}$ can be expressed as

$$\boldsymbol{R}_{m+1}^{-1} \triangleq \begin{bmatrix} \rho_m^{\mathrm{f}} & \boldsymbol{r}_m^{\mathrm{fH}} \\ \boldsymbol{r}_m^{\mathrm{f}} & \boldsymbol{R}_m^{\mathrm{f}} \end{bmatrix}^{-1} = \begin{bmatrix} 0 & \boldsymbol{0}_m^{\mathrm{H}} \\ \boldsymbol{0}_m & (\boldsymbol{R}_m^{\mathrm{f}})^{-1} \end{bmatrix} + \frac{1}{\alpha_m^{\mathrm{f}}}\begin{bmatrix} 1 \\ \boldsymbol{a}_m \end{bmatrix}\begin{bmatrix} 1 & \boldsymbol{a}_m^{\mathrm{H}} \end{bmatrix} \tag{6.1.26}$$

where

$$\boldsymbol{a}_m \triangleq [a_1^{(m)}\ a_2^{(m)}\ \cdots\ a_m^{(m)}]^{\mathrm{T}} \triangleq -(\boldsymbol{R}_m^{\mathrm{f}})^{-1}\boldsymbol{r}_m^{\mathrm{f}} \tag{6.1.27}$$

$$\alpha_m^{\mathrm{f}} \triangleq \rho_m^{\mathrm{f}} - \boldsymbol{r}_m^{\mathrm{f}H}(\boldsymbol{R}_m^{\mathrm{f}})^{-1}\boldsymbol{r}_m^{\mathrm{f}} = \rho_m^{\mathrm{f}} + \boldsymbol{r}_m^{\mathrm{f}H}\boldsymbol{a}_m = \frac{\det \boldsymbol{R}_{m+1}}{\det \boldsymbol{R}_m^{\mathrm{f}}} \tag{6.1.28}$$

and the relationship (6.1.26) exists if matrix $\boldsymbol{R}_m^f$ is invertible and $\alpha_m^f \neq 0$. A similar set of formulas can be obtained for arbitrary matrices (see Problem 6.3).

Interpretations. The vector $\boldsymbol{b}_m$, defined by (6.1.22), is the MMSE estimator of observation x_{m+1} from data vector $\boldsymbol{x}_m$. Indeed, if

$$e_m^{\mathrm{b}} = x_{m+1} - \hat{x}_{m+1} = x_{m+1} + \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{x}_m \tag{6.1.29}$$

we can show, using the orthogonality principle $E\{\boldsymbol{x}_m e_m^{\mathrm{b}*}\} = 0$, that $\boldsymbol{b}_m$ results in the MMSE given by

$$P_m^{\mathrm{b}} = \rho_m^{\mathrm{b}} + \boldsymbol{b}_m^H \boldsymbol{r}_m^{\mathrm{b}} = \alpha_m^{\mathrm{b}} \tag{6.1.30}$$

Similarly, we can show that $\boldsymbol{a}_m$, defined by (6.1.27), is the optimum estimator of x_1 based on $\tilde{\boldsymbol{x}}_m \triangleq [x_2\ x_3\ \cdots\ x_{m+1}]^{\mathrm{T}}$. By using the orthogonality principle, $E\{\boldsymbol{x}_m e_m^{f*}\} = 0$, the MMSE is

$$P_m^{\mathrm{f}} = \rho_m^{\mathrm{f}} + \boldsymbol{r}_m^{\mathrm{fH}}\boldsymbol{a}_m = \alpha_m^{\mathrm{f}} \tag{6.1.31}$$

If $\boldsymbol{x}_{m+1} = [x(n)\ x\ (n-1)\ \cdots\ x(n-m)]^{\mathrm{T}}$, then $\boldsymbol{b}_m$ provides the *backward linear predictor* (*BLP*) and $\boldsymbol{a}_m$ the *forward linear predictor* (*FLP*) of the process $x(n)$ from Section 5.5. For convenience, we always use this terminology even if, strictly speaking, the linear prediction interpretation is not applicable.

6.1.3 Levinson Recursion for the Optimum Estimator

We now illustrate how to use (6.1.24) to express the optimum estimator $\boldsymbol{c}_{m+1}$ in terms of the estimator $\boldsymbol{c}_m$. Indeed, using (6.1.24), (6.1.10), and the normal equations $\boldsymbol{R}_m\boldsymbol{c}_m = \boldsymbol{d}_m$, we have

$$\begin{aligned} \boldsymbol{c}_{m+1} &= \boldsymbol{R}_{m+1}^{-1}\boldsymbol{d}_{m+1} \\ &= \begin{bmatrix} \boldsymbol{R}_m^{-1} & \boldsymbol{0}_m \\ \boldsymbol{0}_m^T & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{d}_m \\ d_{m+1} \end{bmatrix} + 1/\alpha_m^b \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{b}_m^{\mathrm{H}} & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{d}_m \\ d_{m+1} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{R}_m^{-1}\boldsymbol{d}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix}\frac{\boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{d}_m + d_{m+1}}{\alpha_m^b} \end{aligned}$$

or more concisely

$$\boldsymbol{c}_{m+1} = \begin{bmatrix} \boldsymbol{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix} k_m^c \tag{6.1.32}$$

where the quantities

$$k_m^c \triangleq \frac{\beta_m^c}{\alpha_m^b} \tag{6.1.33}$$

and

$$\beta_m^c \triangleq \boldsymbol{b}_m^{\mathrm{H}} \boldsymbol{d}_m + d_{m+1} \tag{6.1.34}$$

contain the "new information" d_{m+1} (the new component of $\boldsymbol{d}_{m+1}$). By using (6.1.22) and $\boldsymbol{R}_m \boldsymbol{c}_m = \boldsymbol{d}_m$, alternatively β_m^c can be written as

$$\beta_m^c = -\boldsymbol{r}_m^{b\mathrm{H}} \boldsymbol{c}_m + d_{m+1} \tag{6.1.35}$$

We will use the term *Levinson recursion* for the order-updating relation (6.1.32) because a similar recursion was introduced as part of the celebrated algorithm due to Levinson (see Section 6.3). However, we stress that even though (6.1.32) is order-recursive, the parameter vector $\boldsymbol{c}_{m+1}$ *does not* have the optimum nesting property, that is, $\boldsymbol{c}_{m+1}^{\lceil m \rceil} \neq \boldsymbol{c}_m$.

Clearly, if we know the vector $\boldsymbol{b}_m$, we can determine $\mathbf{c}_{m+1}$, using (6.1.32); however, its practical utility depends on how easily we can obtain the vector $\boldsymbol{b}_m$. In general, $\boldsymbol{b}_m$ requires the solution of an $m \times m$ linear system of equations, and the computational savings compared to direct solution of the $(m+1)$st-order normal equations is insignificant. For the Levinson recursion to be useful, we need an order recursion for vector $\boldsymbol{b}_m$. Since matrix $\boldsymbol{R}_{m+1}$ has the optimum nesting property, we need to check whether the same is true for the right-hand side vector in $\boldsymbol{R}_{m+1} \boldsymbol{b}_{m+1} = -\boldsymbol{r}_{m+1}^b$. From the definition $\boldsymbol{r}_m^b \triangleq E\{\boldsymbol{x}_m x_{m+1}^*\}$, we can easily see that $\boldsymbol{r}_{m+1}^{b\lceil m \rceil} \neq \boldsymbol{r}_m^b$ and $\boldsymbol{r}_{m+1}^{b\lfloor m \rfloor} \neq \boldsymbol{r}_m^b$. Hence, in general, we cannot find a Levinson recursion for vector $\boldsymbol{b}_m$. This is possible *only* in optimum filtering problems in which the input data vector $\mathbf{x}_m(n)$ has a shift-invariance structure (see Section 6.3).

EXAMPLE 6.1.2 Use the Levinson recursion to determine the optimum linear estimator $\mathbf{c}_3$ specified by the matrix

$$\boldsymbol{R}_3 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix}$$

in Example 6.1.1 and the cross-correlation vector

$$\boldsymbol{d}_3 = [1\ 2\ 4]^{\mathrm{T}}$$

Solution. For $m=1$ we have $r_{11} c_1^{(1)} = d_1$, which gives $c_1^{(1)} = 1$. Also, from (6.1.32) and (6.1.34) we obtain $k_0^c = c_1^{(1)} = 1$ and $\beta_0^c = d_1 = 1$. Finally, from $k_0^c = \beta_0^c / \alpha_0^b$, we get $\alpha_0^b = 1$.

To obtain $\boldsymbol{c}_2$, we need $b_1^{(1)}$, k_1^c, β_1^c, and α_1. We have

$$r_{11} b_1^{(1)} = -r_1^b \Rightarrow b_1^{(1)} = -\frac{\frac{1}{2}}{1} = -\frac{1}{2}$$

$$\beta_1^c = b_1^{(1)} d_1 + d_2 = -\frac{1}{2}(1) + 2 = \frac{3}{2}$$

$$\alpha_1^b = \rho_1^b + r_1^b b_1^{(1)} = 1 + \frac{1}{2}(-\frac{1}{2}) = \frac{3}{4}$$

$$k_1^c = \frac{\beta_1^c}{\alpha_1^b} = 2$$

and therefore

$$\boldsymbol{c}_2 = \begin{bmatrix} \boldsymbol{c}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_1 \\ 1 \end{bmatrix} k_1^c = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} 2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

To determine c_3, we need b_2, β_2^c, and α_2^b. To obtain $\mathbf{b}_2$, we solve the linear system

$$R_2 b_2 = -r_2^b \qquad \text{or} \qquad \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} = -\begin{bmatrix} \frac{1}{3} \\ \frac{1}{2} \end{bmatrix} \Rightarrow b_2 = -\frac{1}{9}\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

and then compute

$$\beta_2^c = b_2^T d_2 + d_3 = -\frac{1}{9}[1 \quad 4]\begin{bmatrix} 1 \\ 2 \end{bmatrix} + 4 = 3$$

$$\alpha_2^b = \rho_2^b + r_2^{bT} b_2 = 1 + \begin{bmatrix} \frac{1}{3} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 \\ 4 \end{bmatrix}\left(-\frac{1}{9}\right) = \frac{20}{27}$$

$$k_2^c = \beta_2^c / \alpha_2^b = \frac{81}{20}$$

The desired solution c_3 is obtained by using the Levinson recursion

$$c_3 = \begin{bmatrix} c_2 \\ 0 \end{bmatrix} + \begin{bmatrix} b_2 \\ 1 \end{bmatrix} k_2^c \Rightarrow c_3 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{9} \\ -\frac{4}{9} \\ 1 \end{bmatrix} \frac{81}{20} = \frac{1}{20}\begin{bmatrix} -9 \\ 4 \\ 81 \end{bmatrix}$$

which agrees with the solution obtained by solving $R_3 c_3 = d_3$ using the function `c3=R3\d3`. We can also solve this linear system by developing an algorithm using the lower partitioning (6.1.26) as discussed in Problem 6.4.

Matrix inversion and the linear system solution for $m = 1$ are trivial (scalar division only). If R_M is strictly positive definite, that is, $R_m = R_M^{[m]}$ is positive definite for all $1 \le m \le M$, the inverse matrices R_m^{-1} and the solutions of $R_m c_m = d_m$, $2 \le m \le M$, can be determined using (6.1.22) and the Levinson recursion (6.1.32) for $m = 1, 2, \cdots, M-1$. However, in practice using the LDL^{H} provides a better method for performing these computations.

6.1.4 Order-Recursive Computation of the LDL^{H} Decomposition

We start by showing that the LDL^H decomposition can be computed in an order-recursive manner. The procedure is developed as part of a formal proof of the LDL^H decomposition using induction.

For $M = 1$, the matrix R_1 is a positive number r_{11} and can be written uniquely in the form $r_{11} = 1 \cdot \xi_1 \cdot 1 > 0$. As we increment the order m, the $(m+1)$st-order principal submatrix of R_m can be partitioned as in (6.1.6). By the induction hypothesis, there are unique matrices L_m and D_m such that

$$R_m = L_m D_m L_m^{\text{H}} \tag{6.1.36}$$

We next form the matrices

$$L_{m+1} = \begin{bmatrix} L_m & 0 \\ l_m^{\text{H}} & 1 \end{bmatrix} \qquad D_{m+1} = \begin{bmatrix} D_m & 0 \\ 0^{\text{H}} & \xi_{m+1} \end{bmatrix} \tag{6.1.37}$$

and try to determine the vector l_m and the positive number ξ_{m+1} so that

$$R_{m+1} = L_{m+1} D_{m+1} L_{m+1}^{\text{H}} \tag{6.1.38}$$

Using (6.1.6) and (6.1.36) through (6.1.38), we see that

$$(L_m D_m) l_m = r_m^b \tag{6.1.39}$$

$$\rho_m^b = l_m^{\text{H}} D_m l_m + \xi_{m+1}, \qquad \xi_{m+1} > 0 \tag{6.1.40}$$

Since
$$\det \boldsymbol{R}_m = \det \boldsymbol{L}_m \det \boldsymbol{D}_m \det \boldsymbol{L}_m^{\mathrm{H}} = \xi_1 \xi_2 \cdots \xi_m > 0 \tag{6.1.41}$$
then $\det \boldsymbol{L}_m \boldsymbol{D}_m \neq 0$ and (6.1.39) has a unique solution $\boldsymbol{l}_m$ Finally, from (6.1.41) we obtain $\xi_{m+1} = \det \boldsymbol{R}_{m+1} / \det \boldsymbol{R}_m$, and therefore $\xi_{m+1} > 0$ because $\boldsymbol{R}_{m+1}$ is positive definite. Hence, ξ_{m+1} is uniquely computed from (6.1.41), which completes the proof.

Because the triangular matrix $\boldsymbol{L}_m$ is generated row by row using (6.1.39) and because the diagonal elements of matrix $\boldsymbol{D}_m$ are computed sequentially using (6.1.40), both matrices have the optimum nesting property, that is, $\boldsymbol{L}_m = \boldsymbol{L}^{\lceil m \rceil}$, $\boldsymbol{D}_m = \boldsymbol{D}^{\lceil m \rceil}$. The optimum filter $\mathbf{c}_m$ is then computed by solving

$$\boldsymbol{L}_m \boldsymbol{D}_m \boldsymbol{k}_m \triangleq \boldsymbol{d}_m \tag{6.1.42}$$

$$\boldsymbol{L}_m^{\mathrm{H}} \boldsymbol{c}_m = \boldsymbol{k}_m \tag{6.1.43}$$

Using (6.1.42), we can easily see that $\boldsymbol{k}_m$ has the optimum nesting property, that is, $\boldsymbol{k}_m = \boldsymbol{k}^{\lceil m \rceil}$ for $1 \leq m \leq M$. This is a consequence of the lower triangular form of $\boldsymbol{L}_m$. The computation of $\boldsymbol{L}_m$, $\boldsymbol{D}_m$, and $\boldsymbol{k}_m$ can be done in a simple, order-recursive manner, which is all that is needed to compute $\mathbf{c}_m$ for $1 \leq m \leq M$. However, the optimum estimator does not have the optimum nesting property, that is, $\boldsymbol{c}_{m+1}^{\lceil m \rceil} \neq \boldsymbol{c}_m$, because of the backward substitution involved in the solution of the upper triangular system (6.1.43) (see Example 5.3.1).

Using (6.1.42) and (6.1.43), we can write the MMSE for the mth-order linear estimator as

$$P_m = P_y - \boldsymbol{c}_m^H \boldsymbol{d}_m = P_y - \boldsymbol{k}_m^H \boldsymbol{D}_m \boldsymbol{k}_m \tag{6.1.44}$$

which, owing to the optimum nesting property of $\boldsymbol{D}_m$ and $\boldsymbol{k}_m$, leads to

$$P_m = P_{m-1} - \xi_m \, | k_m |^2 \tag{6.1.45}$$

which is initialized with $P_0 = P_y$. Equation (6.1.45) provides an *order-recursive algorithm* for the computation of the MMSE.

6.1.5 Order-Recursive Computation of the Optimum Estimate

The computation of the optimum linear estimate $\hat{y}_m = \boldsymbol{c}_m^{\mathrm{H}} \boldsymbol{x}_m$, using a linear combiner, requires m multiplications and $m-1$ additions. Therefore, if we want to compute $\hat{y}_m$, for $1 \leq m \leq M$, we need M linear combiners and hence $M(M+1)/2$ operations.

We next provide an alternative, more efficient order-recursive implementation that exploits the triangular decomposition of $\boldsymbol{R}_{m+1}$. We first notice that using (6.1.43), we obtain

$$\hat{y}_m = \boldsymbol{c}_m^{\mathrm{H}} \boldsymbol{x}_m = (\boldsymbol{k}_m^{\mathrm{H}} \boldsymbol{L}_m^{-1}) \boldsymbol{x}_m = \boldsymbol{k}_m^{\mathrm{H}} (\boldsymbol{L}_m^{-1} \boldsymbol{x}_m) \tag{6.1.46}$$

Next, we define vector $\boldsymbol{w}_m$ as

$$\boldsymbol{L}_m \boldsymbol{w}_m \triangleq \boldsymbol{x}_m \tag{6.1.47}$$

which can be found by using forward substitution in order to solve the triangular system. Therefore, we obtain

$$\hat{y}_m = \boldsymbol{k}_m^H \boldsymbol{w}_m = \sum_{i=1}^{m} k_i^* \omega_i \tag{6.1.48}$$

which provides the estimate $\hat{y}_m$ in terms of $\mathbf{k}_m$ and $\mathbf{w}_m$, that is, without using the estimator vector $\mathbf{c}_m$. Hence, *if the ultimate goal is the computation of $\hat{y}_m$ we do not need to compute the estimator $\boldsymbol{c}_m$.*

For an order-recursive algorithm to be possible, the vector $\boldsymbol{w}_m$ must have the optimum nesting property, that is, $\boldsymbol{w}_m = \boldsymbol{w}_{m+1}^{\lceil m \rceil}$. Indeed, using (6.1.37) and the matrix inversion by partitioning lemma for nonsymmetric matrices (see Problem 6.3), we obtain

$$L_{m+1}^{-1} = \begin{bmatrix} L_m & 0 \\ l_m^H & 1 \end{bmatrix}^{-1} = \begin{bmatrix} L_m^{-1} & 0 \\ v_m^H & 1 \end{bmatrix}$$

where
$$v_m = -L_m^{-H} l_m = -(L_m^H)^{-1} D_m^{-1} L_m^{-1} r_m^b = -R_m^{-1} r_m^b = b_m$$

due to (6.1.22). Therefore,

$$w_{m+1} = L_{m+1}^{-1} x_{m+1} = \begin{bmatrix} L_m^{-1} & 0 \\ b_m^H & 1 \end{bmatrix} \begin{bmatrix} x_m \\ x_{m+1} \end{bmatrix} = \begin{bmatrix} w_m \\ w_{m+1} \end{bmatrix} \tag{6.1.49}$$

where
$$w_{m+1} = b_m^H x_m + x_{m+1} = e_m^b \tag{6.1.50}$$

from (6.1.29) In this case, we can derive order-recursive algorithms for the computation of $\hat{y}_m$ and e_m, for all $1 \le m \le M$. Indeed, using (6.1.48) and (6.1.49), we obtain

$$\hat{y}_m = \hat{y}_{m-1} + k_m^* \omega_m \tag{6.1.51}$$

with $\hat{y}_0 = 0$. From (6.1.51) and $e_m = y - \hat{y}_m$, we have

$$e_m = e_{m-1} - k_m^* \omega_m \tag{6.1.52}$$

for $m = 1, 2, \cdots, M$ with $e_0 = y$. The quantity ω_m can be computed in an order-recursive manner by solving (6.1.47) using forward substitution. Indeed, from the mth row of (6.1.47) we obtain

$$\omega_m = x_m - \sum_{i=1}^{m-1} l_{i-1}^{(m-1)} \omega_i \tag{6.1.53}$$

which provides a *recursive* computation of ω_m for $m = 1, 2, \cdots, M$. To comply with the order-oriented notation, we use $l_{i-1}^{(m-1)}$ instead of $l_{m-1,i-1}$. Depending on the application, we use either (6.1.51) or (6.1.52).

For MMSE estimation, all the quantities are functions of the time index n, and therefore, the triangular decomposition of R_m and the recursions (6.1.51) through (6.1.53) should be repeated for every new set of observations $y(n)$ and $\mathbf{x}(n)$.
A linear estimator is specified by the correlation matrix R_4 and the cross-correlation vector d_4 in

EXAMPLE 6.1.3. A linear estimator is specified by the correlation matrix R_4 and the crosscorrelation vector d_4 are given Compute the estimates $\hat{y}_m$, $1 \le m \le 4$, if the input data vector is given by $x_4 = [1\ 2\ 1\ -1]^T$

.

Solution. Using the triangular factor L4 and the vector k4 found in Example 5.3.2 and (6.1.53),
we find $w_4 = R^{-1} d$ $w_4 = [1 \quad -1 \quad 3 \quad -8]$

and
$$\hat{y}_1 = 1 \qquad \hat{y}_2 = \tfrac{4}{3} \qquad \hat{y}_3 = 6.6 \qquad \hat{y}_4 = 14.6$$

which the reader can verify by computing $\mathbf{c}_m$ and $\hat{y}_m = c_m^T x_m$, $1 \le m \le 4$.

If we compute the matrix

$$B_{m+1} \triangleq L_{m+1}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ b_0^{(1)} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_0^{(m)} & b_1^{(m)} & \cdots & 1 \end{bmatrix} \tag{6.1.54}$$

then (6.1.49) can be written as

$$\boldsymbol{w}_{m+1} = \boldsymbol{e}^b_{m+1} = \boldsymbol{B}_{m+1}\boldsymbol{x}_{m+1} \tag{6.1.55}$$

where

$$e^b_{m+1} \triangleq [e^b_0 \; e^b_1 \; \cdots \; e^b_m]^T \tag{6.1.56}$$

is the BLP error vector. From (6.1.22), we can easily see that the rows of $\boldsymbol{B}_{m+1}$ are formed by the optimum estimators $\boldsymbol{b}_m$ of x_{m+1} from $\mathbf{x}_m$. Note that the elements of matrix $\boldsymbol{B}_{m+1}$ are denoted by using the order-oriented notation $b_i^{(m)}$ introduced in Section 6.1 rather than the conventional b_{mi} matrix notation. Equation (6.1.55) provides an alternative computation of $\mathbf{w}_{m+1}$ as a matrix-vector multiplication. Each component of $\mathbf{w}_{m+1}$ can be computed independently, and hence in parallel, by the formula

$$\omega_j = x_j + \sum_{i=1}^{j-1} b_{i-1}^{(j-1)*} x_i \qquad 1 \le j \le m \tag{6.1.57}$$

which, in contrast to (6.1.53), is nonrecursive. Using (6.1.57) and (6.1.51), we can derive the order-recursive MMSE estimator implementation shown in Figure 6.1.

Finally, we notice that matrix $\boldsymbol{B}_m$ provides the UDU^{H} decomposition of the inverse correlation matrix $\boldsymbol{R}_m$. Indeed, from (6.1.36) we obtain

$$\boldsymbol{R}_m^{-1} = (\boldsymbol{L}_m^{\mathrm{H}})^{-1}\boldsymbol{D}_m^{-1}\boldsymbol{L}_m^{-1} = \boldsymbol{B}_m^{\mathrm{H}}\boldsymbol{D}_m^{-1}\boldsymbol{B}_m \tag{6.1.58}$$

because inversion and transposition are interchangeable and the UDU^H decomposition is unique. This formula provides a practical method to compute the inverse of the correlation matrix by using the $\boldsymbol{LDL}^{\mathrm{H}}$ decomposition because computing the inverse of a triangular matrix is simple (see Problem 6.5).

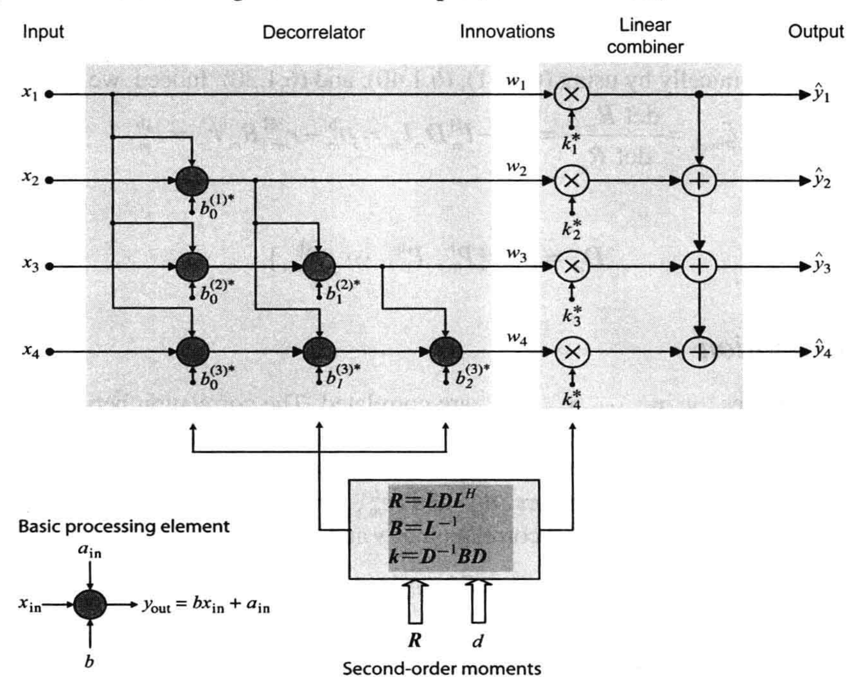


FIGURE 6.1
Orthogonal order-recursive structure for linear MMSE estimation.

6.2 Interpretations of Algorithmic Quantities

We next show that various intermediate quantities that appear in the linear MMSE estimation algorithms have physical and statistical interpretations that, besides their intellectual value, facilitate better understanding of the

operation, performance, and numerical properties of the algorithms.

6.2.1 Innovations and Backward Prediction

The correlation matrix of $\boldsymbol{w}_m$ is

$$E\{\boldsymbol{w}_m\boldsymbol{w}_m^{\mathrm{H}}\} = \boldsymbol{L}_m^{-1}E\{\boldsymbol{x}_m\boldsymbol{x}_m^{\mathrm{H}}\}\boldsymbol{L}_m^{-\mathrm{H}} = \boldsymbol{D}_m \tag{6.2.1}$$

where we have used (6.1.47) and the triangular decomposition (6.1.36). Therefore, the components of $\boldsymbol{w}_m$ are uncorrelated, random variables with variances

$$\xi_i = E\{|\omega_i|^2\} \tag{6.2.2}$$

since $\xi_i \geq 0$. Furthermore, the two sets of random variables $\{\omega_1, \omega_2, ..., \omega_M\}$ and $\{x_1, x_2, \cdots, x_M\}$ are *linearly equivalent* because they can be obtained from each other through the linear transformation (6.1.47). This transformation removes all the redundant correlation among the components of $\mathbf{x}$ and is known as a *decorrelation* or *whitening* operation (see Section 3.3.2). Because the random variables ω_i are uncorrelated, each of them adds "new information" or innovation. In this sense, $\{\omega_1, \omega_2, \cdots, \omega_m\}$ is the *innovations representation* of the random variables $\{x_1, x_2, \cdots, x_m\}$. Because $\boldsymbol{x}_m = \boldsymbol{L}_m\boldsymbol{w}_m$, the random vector $\boldsymbol{w}_m = \boldsymbol{e}_m^b$ is the *innovations representation*, and $\boldsymbol{x}_m$ and $\boldsymbol{w}_m$ are *linearly equivalent* as well,

The cross-correlation matrix between $\boldsymbol{x}_m$ and $\boldsymbol{w}_m$ is

$$E\{\boldsymbol{x}_m\boldsymbol{w}_m^{\mathrm{H}}\} = E\{\boldsymbol{L}_m\boldsymbol{w}_m\boldsymbol{w}_m^{\mathrm{H}}\} = \boldsymbol{L}_m\boldsymbol{D}_m \tag{6.2.3}$$

which shows that, owing to the lower triangular form of $\boldsymbol{L}_m$, $E\{x_i\omega_j^*\} = 0$ for $j > i$.

Furthermore, since $e_m^{\mathrm{b}} = \omega_{m+1}$, from (6.1.50) we have

$$P_m^{\mathrm{b}} = \xi_{m+1} = E\{|\omega_{m+1}|^2\}$$

which also can be shown algebraically by using (6.1.41), (6.1.40), and (6.1.30). Indeed, we have

$$\xi_{m+1} = \frac{\det \boldsymbol{R}_{m+1}}{\det \boldsymbol{R}_m} = \rho_m^{\mathrm{b}} - \boldsymbol{l}_m^{\mathrm{H}}\boldsymbol{D}_m\boldsymbol{l}_m = \rho_m^{\mathrm{b}} - \boldsymbol{r}_m^{\mathrm{bH}}\boldsymbol{R}_m^{-1}\boldsymbol{r}_m^{\mathrm{b}} = P_m^{\mathrm{b}} \tag{6.2.4}$$

and, therefore,

$$\boldsymbol{D}_m = \mathrm{diag}\{P_0^{\mathrm{b}}, P_1^{\mathrm{b}}, \cdots, P_{m-1}^{\mathrm{b}}\} \tag{6.2.5}$$

6.2.2 Partial Correlation

In general, the random variables $y, x_1, \cdots, x_m, x_{m+1}$ are correlated. The correlation between y and x_{m+1}, after the influence from the components of the vector $\boldsymbol{x}_m$ has been removed, is known as *partial correlation*. To remove the correlation due to $\boldsymbol{x}_m$, we extract from y and x_{m+1} the components that can be predicted from $\boldsymbol{x}_m$. The remaining correlation is from the estimation errors e_m and e_m^b, which are both uncorrelated with $\boldsymbol{x}_m$ because of the orthogonality principle. Therefore, the partial correlation of y and x_{m+1} is

$$\begin{aligned}\mathrm{PARCOR}(y;\, x_{m+1}) &\triangleq E\{e_m e_m^{\mathrm{b}*}\} = E\{(y - \boldsymbol{c}_m^{\mathrm{H}}\boldsymbol{x}_m)e_m^{\mathrm{b}*}\} \\ &= E\{ye_m^{\mathrm{b}*}\} = E\{y(x_{m+1}^* + \boldsymbol{x}_m^{\mathrm{H}}\boldsymbol{b}_m)\} \\ &= E\{yx_{m+1}^*\} + E\{y\boldsymbol{x}_m^{\mathrm{H}}\}\boldsymbol{b}_m \\ &= d_{m+1}^* + \boldsymbol{d}_m^{\mathrm{H}}\boldsymbol{b}_m \triangleq \beta_m^{c*}\end{aligned} \tag{6.2.6}$$

where we have used the orthogonality principle $E\{\boldsymbol{x}_m e_m^{b*}\} = \boldsymbol{0}$ and (6.1.10), (6.1.50), and (6.1.34).

The partial correlation $\mathrm{PARCOR}(y;\, x_{m+1})$ is also related to the parameters k_m obtained from the LDL^{H} decomposition. Indeed, from (6.1.42) and (6.1.54), we obtain the relation

$$\boldsymbol{k}_{m+1} = \boldsymbol{D}_{m+1}^{-1}\boldsymbol{B}_{m+1}\boldsymbol{d}_{m+1} \tag{6.2.7}$$

whose last row is

$$k_{m+1} = \frac{b_m^H d_m + d_{m+1}}{\xi_{m+1}} = \beta_m^c / P_m^b = k_m^c \tag{6.2.8}$$

owing to (6.2.4) and (6.2.6).

EXAMPLE 6.2.1 The LDL^H decomposition of matrix R_3 in Example 6.1.2 is given by

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & \frac{4}{9} & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{4} & 0 \\ 0 & 0 & \frac{20}{27} \end{bmatrix}$$

and can be found by using the function `[L,D]=ldlt(R)`. Comparison with the results obtained in Example 6.1.2 shows that the rows of the matrix

$$L^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{9} & -\frac{4}{9} & 1 \end{bmatrix}$$

provide the elements of the backward predictors, whereas the diagonal elements of D are equal to the scalars α_m. Using (6.2.7), we obtain $k = [1\ 2\ \frac{81}{20}]^T$ whose elements are the quantities k_0^c, k_1^c, and k_2^c computed in Example 6.1.2 using the Levinson recursion.

6.2.3 *Order Decomposition of the Optimum Estimate*

The equation $\hat{y}_{m+1} = \hat{y}_m + k_{m+1}^* \omega_{m+1}$, with $k_{m+1} = \beta_m^c / P_m^b = k_m^c$, shows that the improvement in the estimate when we include one more observation x_{m+1}, that is, when we increase the order by 1, is proportional to the innovation ω_{m+1} contained in x_{m+1}. The innovation is the part of x_{m+1} that cannot be linearly estimated from the already used data $\boldsymbol{x}_m$. The term ω_{m+1} is scaled by the ratio of the partial correlation between y and the "new" observation x_{m+1} and the power of the innovation P_m^b.

Thus, *the computation of the* $(m+1)$ *st-order estimate of y based on* $\boldsymbol{x}_{m+1} = [\boldsymbol{x}_m^T\ x_{m+1}]$ *can be reduced to two mth-order estimation problems: the estimation of* y *based on* $\boldsymbol{x}_m$ *and the estimation of the new observation* x_{m+1} *based on* $\boldsymbol{x}_m$. This decomposition of linear estimation problems into smaller ones has very important applications to the development of efficient algorithms and structures for MMSE estimation.

We use the term *direct* for the implementation of the MMSE linear combiner as a sum of products, involving the optimum parameters $c_i^{(m)}, 1 \le i \le m$, to emphasize the direct use of these coefficients. Because the random variables ω_i used in the implementation of Figure 6.1 are orthogonal, that is, $<\omega_i, \omega_j> = 0$ for $i \ne j$, we refer to this implementation as the *orthogonal implementation* or the *orthogonal structure*. These two structures appear in every type of linear MMSE estimation problem, and their particular form depends on the specifics of the problem and the associated second-order moments. In this sense, they play a prominent role in linear MMSE estimation in general, and in this book in particular.

We conclude our discussion with the following important observations:

1. The direct implementation combines correlated, that is, redundant information, and it is *not* order-recursive because increasing the order of the estimator *destroys* the optimality of the existing coefficients. Again, the reason is that the direct-form optimum filter coefficients do not possess the optimal nesting property.
2. The orthogonal implementation consists of a decorrelator and a linear combiner. The estimator combines the innovations of the data (nonredundant information) and is order-recursive because it does not use the optimum coefficient vector. Hence, increasing the order of the estimator preserves the optimality of the existing

lower-order part. The resulting structure is modular such that each additional term improves the estimate by an amount proportional to the included innovation ω_m.

3. Using the vector interpretation of random variables, the transformation $\tilde{\mathbf{x}}_m = \mathbf{F}_m \mathbf{x}_m$ is just a change of basis. The choice $\mathbf{F}_m = \mathbf{L}_m^{-1}$ converts from the *oblique* set $\{x_1, x_2, \cdots, x_m\}$ to the *orthogonal* basis $\{\omega_1, \omega_2, \cdots, \omega_m\}$. The advantage of working with orthogonal bases is that adding new components does not affect the optimality of previous ones.
4. The LDL^H decomposition for random vectors is the matrix equivalent of the spectral factorization theorem for discrete-time, stationary, stochastic processes. Both approaches facilitate the design and implementation of optimum FIR and IIR filters (see Sections 5.3 and 5.6).

6.2.4 Gram-Schmidt Orthogonalization

We next combine the geometric interpretation of the random variables with the Gram-Schmidt procedure used in linear algebra. The Gram-Schmidt procedure produces the innovations $\{\omega_1, \omega_2, \cdots, \omega_m\}$ by orthogonalizing the original set $\{x_1, x_2, \cdots, x_m\}$.

We start by choosing ω_1 to be in the direction of x_1, that is,

$$\omega_1 = x_1$$

The next "vector" ω_2 should be orthogonal to ω_1. To determine ω_2, we subtract from x_2 its component along ω_1 [see Figure 6.2(*a*)], that is,

$$\omega_2 = x_2 - l_0^{(1)}\omega_1$$

where $l_0^{(1)}$ is obtained from the condition $\omega_2 \perp \omega_1$ as follows:

$$< \omega_2, \omega_1 > \; = \; < x_2, \omega_1 > - l_0^{(1)} < \omega_1, \omega_1 > \; = \; 0$$

or

$$l_0^{(1)} = \frac{< x_2, \omega_1 >}{< \omega_1, \omega_1 >}$$

Similarly, to determine ω_3, we subtract from x_3 its components along ω_1 and ω_2, that is,

$$\omega_3 = x_3 - l_0^{(2)}\omega_1 - l_1^{(2)}\omega_2$$

as illustrated in Figure 6.2(*b*). Using the conditions $\omega_3 \perp \omega_1$ and $\omega_3 \perp \omega_2$, we can easily see that

$$l_0^{(2)} = \frac{< x_3, \omega_1 >}{< \omega_1, \omega_1 >} \qquad l_1^{(2)} = \frac{< x_3, \omega_2 >}{< \omega_2, \omega_2 >}$$

This approach leads to the following *classical Gram-Schmidt algorithm*:

- Define $\omega_1 = x_1$.
- For $2 \le m \le M$, compute

$$\omega_m = x_m - l_0^{(m-1)}\omega_1 \cdots - l_{m-2}^{(m-1)}\omega_{m-1} \tag{6.2.9}$$

where

$$l_i^{(m-1)} = \frac{< x_{m-1}, \omega_i >}{< \omega_i, \omega_i >} \tag{6.2.10}$$

assuming that $< \omega_i, \omega_i > \; \neq 0$.

From the derivation of the algorithm it should be clear that the sets $\{x_1, \cdots, x_m\}$ and $\{\omega_1, \cdots, \omega_m\}$ are linearly equivalent for $m = 1, 2, \cdots, M$. Using (6.2.11), we obtain

$$\mathbf{x}_m = \mathbf{L}_m \mathbf{w}_m \tag{6.2.11}$$

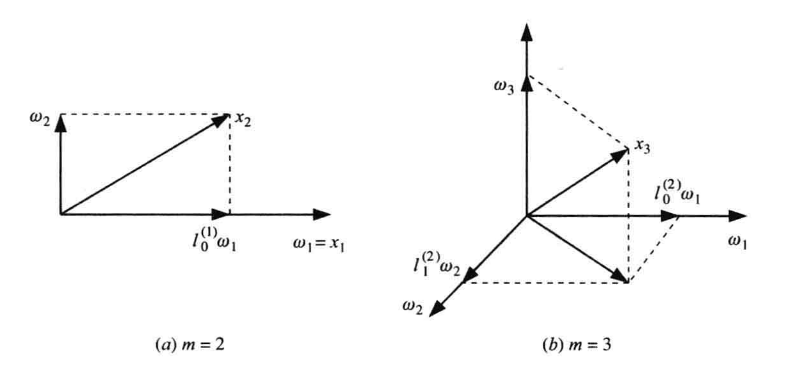


FIGURE 6.2
Illustration of the Gram-Schmidt orthogonalization process.

where

$$\boldsymbol{L}_m \triangleq \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_0^{(1)} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_0^{(m-1)} & l_1^{(m-1)} & \cdots & 1 \end{bmatrix} \tag{6.2.12}$$

is a unit lower triangular matrix. Since, by construction, the components of $\boldsymbol{w}_m$ are uncorrelated, its correlation matrix $\boldsymbol{D}_m$ is diagonal with elements $\xi_i = E\{|\omega_i|^2\}$. Using (6.2.11), we obtain

$$\boldsymbol{R}_m = E\{\boldsymbol{x}_m\boldsymbol{x}_m^H\} = \boldsymbol{L}_m E\{\boldsymbol{w}_m\boldsymbol{w}_m^H\}\boldsymbol{L}_m^H = \boldsymbol{L}_m\boldsymbol{D}_m\boldsymbol{L}_m^H \tag{6.2.13}$$

which is precisely the unique $\boldsymbol{LDL}^H$ decomposition of the correlation matrix $\boldsymbol{R}_m$. Therefore, *the Gram-Schmidt orthogonalization of the data vector* $\boldsymbol{x}_m$ *provides an alternative approach to obtain the* LDL^H *decomposition of its correlation matrix* $\boldsymbol{R}_m = E\{\boldsymbol{x}_m\boldsymbol{x}_m^H\}$.

6.3 Order-Recursive Algorithms for Optimum FIR Filters

The key difference between a linear combiner and an FIR filter is the nature of the input data vector. The input data vector for FIR filters consists of *consecutive samples* from the *same* discrete-time stochastic process, that is,

$$\boldsymbol{x}_m(n) = [x(n)\ x\ (n-1)\ \cdots\ x(n-m+1)]^{\mathrm{T}} \tag{6.3.1}$$

instead of samples from m different processes $x_i(n)$. This *shift invariance* of the input data vector allows for the development of simpler, order-recursive algorithms and structures for optimum FIR filtering and prediction compared to those for general linear estimation. Furthermore, the quest for order-recursive algorithms leads to a natural, elegant, and unavoidable interconnection between optimum filtering and the BLP and FLP problems.

We start with the following upper and lower partitioning of the input data vector

$$\boldsymbol{x}_{m+1}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-m+1) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \boldsymbol{x}_m(n-1) \end{bmatrix} \tag{6.3.2}$$

which shows that $\boldsymbol{x}_{m+1}^{\lceil m \rceil}(n)$ and $\boldsymbol{x}_{m+1}^{\lfloor m \rfloor}(n)$ are simply shifted versions (by one sample delay) of the same vector $\boldsymbol{x}_m(n)$. The shift invariance of $\boldsymbol{x}_{m+1}(n)$ results in an analogous shift invariance for the correlation matrix $\boldsymbol{R}_{m+1}(n) = E\{\boldsymbol{x}_{m+1}(n)\boldsymbol{x}_{m+1}^H(n)\}$. Indeed, we can easily show that the upper-lower partitioning of the correlation matrix is

$$\boldsymbol{R}_{m+1}(n) = \begin{bmatrix} \boldsymbol{R}_m(n) & \boldsymbol{r}_m^{\mathrm{b}}(n) \\ \boldsymbol{r}_m^{\mathrm{bH}}(n) & P_x(n-m) \end{bmatrix} \tag{6.3.3}$$

and the lower-upper partitioning is

$$\boldsymbol{R}_{m+1}(n) = \begin{bmatrix} P_x(n) & \boldsymbol{r}_m^{\mathrm{fH}}(n) \\ \boldsymbol{r}_m^{\mathrm{f}}(n) & \boldsymbol{R}_m(n-1) \end{bmatrix} \tag{6.3.4}$$

where
$$\boldsymbol{r}_m^{\mathrm{b}}(n) = E\{\boldsymbol{x}_m(n)x^*(n-m)t\} \tag{6.3.5}$$

$$\boldsymbol{r}_m^{\mathrm{f}}(n) = E\{\boldsymbol{x}_m(n-1)x^*(n)\} \tag{6.3.6}$$

$$P_x(n) = E\{|\, x(n)\,|^2\} \tag{6.3.7}$$

We note that, in contrast to the general case (6.1.5) where the matrix $\boldsymbol{R}_m^f(n) = \boldsymbol{R}_{m+1}^{\lfloor m \rfloor}(n)$ is *unrelated* to $\boldsymbol{R}_m(n)$, here the matrix $\boldsymbol{R}_{m+1}^{\lfloor m \rfloor}(n) = \boldsymbol{R}_m(n-1)$. This is a by-product of the shift-invariance property of the input data vector and takes the development of order-recursive algorithms one step further. We begin our pursuit of an order-recursive algorithm with the development of a Levinson order recursion for the optimum FIR filter coefficients.

6.3.1 Order-Recursive Computation of the Optimum Filter

Suppose that at time n we have already computed the optimum FIR filter $\mathbf{c}_m(n)$ specified by

$$\boldsymbol{c}_m(n) = \boldsymbol{R}_m^{-1}(n)\boldsymbol{d}_m(n) \tag{6.3.8}$$

and the MMSE is

$$P_m^c(n) = P_y(n) - \boldsymbol{d}_m^{\mathrm{H}}(n)\boldsymbol{c}_m(n) \tag{6.3.9}$$

where
$$\mathrm{d}_m(n) = E\{\boldsymbol{x}_m(n)y^*(n)\} \tag{6.3.10}$$

We wish to compute the optimum filter

$$\boldsymbol{c}_{m+1}(n) = \boldsymbol{R}_{m+1}^{-1}(n)\boldsymbol{c}_{m+1}(n)$$

by modifying $\boldsymbol{c}_m(n)$ using an order-recursive algorithm. From (6.3.3), we see that matrix $\boldsymbol{R}_{m+1}(n)$ has the optimum nesting property. Using the upper partitioning in (6.3.2), we obtain

$$\boldsymbol{d}_{m+1}(n) = E\left\{\begin{bmatrix} \boldsymbol{x}_m(n) \\ x(n-m) \end{bmatrix} y^*(n)\right\} = \begin{bmatrix} \boldsymbol{d}_m(n) \\ d_{m+1}(n) \end{bmatrix} \tag{6.3.11}$$

which shows that $\boldsymbol{d}_{m+1}(n)$ also has the optimum nesting property. Therefore, we can develop a Levinson order recursion using the upper left matrix inversion by partitioning lemma

$$\boldsymbol{R}_{m+1}^{-1}(n) = \begin{bmatrix} \boldsymbol{R}_m^{-1}(n) & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix} + \frac{1}{P_m^b(n)} \begin{bmatrix} \boldsymbol{b}_m(n) \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_m^{\mathrm{H}}(n) & 1 \end{bmatrix} \tag{6.3.12}$$

where
$$\boldsymbol{b}_m(n) = -\boldsymbol{R}_m^{-1}(n)\boldsymbol{r}_m^b(n) \tag{6.3.13}$$

is the optimum BLP, and

$$P_m^{\mathrm{b}}(n) = \frac{\det \boldsymbol{R}_{m+1}(n)}{\det \boldsymbol{R}_m(n)} = P_x(n-m) + \boldsymbol{r}_m^{\mathrm{bH}}(n)\boldsymbol{b}_m(n) \tag{6.3.14}$$

is the corresponding MMSE. Equations (6.3.12) through (6.3.14) follow easily from (6.1.22), (6.1.23), and (6.1.24). It is interesting to note that $\boldsymbol{b}_m(n)$ is the optimum estimator for the additional observation $x(n-m)$ used by the optimum filter $\boldsymbol{c}_{m+1}(n)$. Substituting (6.3.11) and (6.3.12) into (6.3.11), we obtain

$$\boldsymbol{c}_{m+1}(n) = \begin{bmatrix} \boldsymbol{c}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m(n) \\ 1 \end{bmatrix} k_m^c(n) \tag{6.3.15}$$

where

$$k_m^c(n) \triangleq \frac{\beta_m^c(n)}{P_m^{\mathrm{b}}(n)} \tag{6.3.16}$$

and

$$\beta_m^c(n) \triangleq \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{d}_m(n) + d_{m+1}(n) \tag{6.3.17}$$

Thus, if we know the BLP $\boldsymbol{b}_m(n)$, we can determine $\boldsymbol{c}_{m+1}(n)$ by using the Levinson recursion in (6.3.15).

Levinson recursion for the backward predictor. For the order recursion in (6.3.15) to be useful, we need an order recursion for the BLP $\boldsymbol{b}_m(n)$. This is possible if the linear systems

$$\begin{aligned} \boldsymbol{R}_m(n)\boldsymbol{b}_m(n) &= -\boldsymbol{r}_m^b(n) \\ \boldsymbol{R}_{m+1}(n)\boldsymbol{b}_{m+1}(n) &= -\boldsymbol{r}_{m+1}^b(n) \end{aligned} \tag{6.3.18}$$

are nested. Since the matrices are nested [see (6.3.3)], we check whether the right-hand side vectors are nested. We can easily see that no optimum nesting is possible if we use the upper partitioning in (6.3.2). However, if we use the lower-upper partitioning, we obtain

$$\boldsymbol{r}_{m+1}^{\mathrm{b}}(n) = E\left\{ \begin{bmatrix} x(n) \\ \boldsymbol{x}_m(n-1) \end{bmatrix} x^*(n-m-1) \right\} \triangleq \begin{bmatrix} r_{m+1}^{\mathrm{b}}(n) \\ \boldsymbol{r}_m^{\mathrm{b}}(n-1) \end{bmatrix} \tag{6.3.19}$$

which provides a partitioning that includes the wanted vector $\boldsymbol{r}_m^b(n)$ delayed by one sample as a result of the shift invariance of $\boldsymbol{x}_m(n)$. To explore this partitioning, we use the lower-upper corner matrix inversion by partitioning lemma

$$\boldsymbol{R}_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \boldsymbol{0}^{\mathrm{H}} \\ \boldsymbol{0} & \boldsymbol{R}_m^{-1}(n-1) \end{bmatrix} + \frac{1}{P^f(n)} \begin{bmatrix} 1 \\ \boldsymbol{a}_m(n) \end{bmatrix} \begin{bmatrix} 1 & \boldsymbol{a}_m^{\mathrm{H}}(n) \end{bmatrix} \tag{6.3.20}$$

where

$$\boldsymbol{a}_m(n) \triangleq -\boldsymbol{R}_m^{-1}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n) \tag{6.3.21}$$

is the optimum FLP and

$$P_m^{\mathrm{f}}(n) = \frac{\det \boldsymbol{R}_{m+1}(n)}{\det \boldsymbol{R}_m(n-1)} = P_x(n) + \boldsymbol{r}_m^{\mathrm{fH}}(n)\boldsymbol{a}_m(n) \tag{6.3.22}$$

is the forward linear prediction MMSE. Equations (6.3.20) through (6.3.22) follow easily from (6.1.26) through (6.1.28). Substituting (6.3.20) and (6.3.19) into

$$\boldsymbol{b}_{m+1}(n) = -\boldsymbol{R}_{m+1}^{-1}(n)\boldsymbol{r}_{m+1}^b(n)$$

we obtain the recursion

$$\boldsymbol{b}_{m+1}(n) = \begin{bmatrix} 0 \\ \boldsymbol{b}_m(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \boldsymbol{a}_m(n) \end{bmatrix} k_m^{\mathrm{b}}(n) \tag{6.3.23}$$

where
$$k_m^{\mathrm{b}}(n) \triangleq \frac{-\beta_m^{\mathrm{b}}(n)}{P_m^{\mathrm{f}}(n)} \tag{6.3.24}$$

and
$$\beta_m^{\mathrm{b}}(n) \triangleq r_{m+1}^{\mathrm{b}}(n) + \boldsymbol{a}_m^H(n)\boldsymbol{r}_m^{\mathrm{b}}(n-1) \tag{6.3.25}$$

To proceed with the development of the order-recursive algorithm, we clearly need an order recursion for the optimum FLP $\mathbf{a}_m(n)$.

Levinson recursion for the forward predictor. Following a similar procedure for the Levinson recursion of the BLP, we can derive the Levinson recursion for the FLP. If we use the upper-lower partitioning in (6.3.2), we obtain

$$\boldsymbol{r}_{m+1}^{\mathrm{f}}(n) = E\{\boldsymbol{x}_{m+1}(n-1)x^*(n)\} = \begin{bmatrix} \boldsymbol{r}_m^{\mathrm{f}}(n) \\ r_{m+1}^{\mathrm{f}}(n) \end{bmatrix} \tag{6.3.26}$$

which in conjunction with (6.3.12) and (6.3.21) leads to the following order recursion

$$\boldsymbol{a}_{m+1}(n) = \begin{bmatrix} \boldsymbol{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \tag{6.3.27}$$

where
$$k_m^{\mathrm{f}}(n) \triangleq \frac{-\beta_m^{\mathrm{f}}(n)}{P_m^{\mathrm{b}}(n-1)} \tag{6.3.28}$$

and
$$\beta_m^{\mathrm{f}}(n) \triangleq \boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n) + r_{m+1}^{\mathrm{f}}(n) \tag{6.3.29}$$

Is an order-recursive algorithm feasible? For $m=1$, we have a scalar equation $r_{11}(n)c_1^{(1)}(n) = d_1(n)$ whose solution is $c_1^{(1)}(n) = d_1(n)/r_{11}(n)$. Using the Levinson order recursions for $m=1, 2, \cdots, M-1$, we can find $\boldsymbol{c}_M(n)$ if the quantities $\boldsymbol{b}_m(n-1)$ and $P_m^b(n-1)$, $1 \le m < M$, required by (6.3.27) and (6.3.28) are known. The lack of this information prevents the development of a complete order-recursive algorithm for the solution of the normal equations for optimum FIR filtering or prediction. The need for time updates arises because each order update requires both the upper left corner and the lower right corner partitionings

$$\boldsymbol{R}_{m+1}(n) = \begin{bmatrix} \boldsymbol{R}_m(n) & \times \\ \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times \\ \times & \boldsymbol{R}_m(n-1) \end{bmatrix}$$

of matrix $\boldsymbol{R}_{m+1}$. The presence of $\boldsymbol{R}_m(n-1)$, which is a result of the nonstationarity of the input signal, creates the need for a time updating of $\boldsymbol{b}_m(n)$. This is possible only for certain types of nonstationarity that can be described by simple relations between $\boldsymbol{R}_m(n)$ and $\boldsymbol{R}_m(n-1)$. The simplest case occurs for stationary processes where $\boldsymbol{R}_m(n) = \boldsymbol{R}_m(n-1) = \boldsymbol{R}_m$. Another very useful case occurs for nonstationary processes generated by linear state-space models, which results in the Kalman filtering algorithm (see Section 6.8).

Partial correlation interpretation. The partial correlation between $y(n)$ and $x(n-m)$, after the influence of the intermediate samples $x(n)$, $x(n-1)$, $\cdots$, $x(n-m+1)$ has been removed, is

$$E\{e_m^{\mathrm{b}}(n)e_m^*(n)\} = \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{d}_m(n) + d_{m+1}(n) = \beta_m^c(n) \tag{6.3.30}$$

which is obtained by working as in the derivation of 6.26
It can be shown, that the $k_m(n)$ parameters in the Levinson recursions can be obtained from

$$\begin{aligned} \boldsymbol{R}_m(n) &= \boldsymbol{L}_m(n)\boldsymbol{D}_m(n)\boldsymbol{L}_m^{\mathrm{H}}(n) \\ \boldsymbol{L}_m(n)\boldsymbol{D}_m(n)\boldsymbol{k}_m^c(n) &= \boldsymbol{d}_m(n) \\ \boldsymbol{L}_m(n)\boldsymbol{D}_m(n)\boldsymbol{k}_m^f(n) &= \boldsymbol{r}_m^b(n) \\ \boldsymbol{L}_m(n-1)\boldsymbol{D}_m(n-1)\boldsymbol{k}_m^b(n) &= \boldsymbol{r}_m^f(n) \end{aligned} \tag{6.3.31}$$

that is, as a by-product of the $\mathrm{LDL}^{\mathrm{H}}$ decomposition.

Similarly, if we consider the sequence $x(n), x(n-1), \cdots, x(n-m), x(n-m-1)$, we can show that the partial correlation between $x(n)$ and $x(n-m-1)$ is given by (see Problem 6.6)

$$E\{e_m^{\mathrm{b}}(n-1)e_m^{\mathrm{f}*}(n)\} = r_{m+1}^{\mathrm{f}}(n) + \boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n) = \beta_m^{\mathrm{f}}(n) \tag{6.3.32}$$

Because $r_{m+1}^{\mathrm{f}}(n) = r_{m+1}^{\mathrm{b}*}(n)$, we have the following simplification

$$\begin{aligned}\beta_m^{\mathrm{f}}(n) &= \boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{R}_m(n-1)\boldsymbol{R}_m^{-1}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n) + r_{m+1}^{\mathrm{f}}(n) \\ &= \boldsymbol{r}_m^{\mathrm{bH}}(n-1)\boldsymbol{a}_m(n) + r_{m+1}^{\mathrm{b}*}(n) = \beta_m^{\mathrm{b}*}(n)\end{aligned}$$

which is known as *Burg's lemma* (Burg 1975). In order to simplify the notation, we define

$$\beta_m(n) \triangleq \beta_m^{\mathrm{f}}(n) = \beta_m^{\mathrm{b}*}(n) \tag{6.3.33}$$

Using (6.3.24), (6.3.28), and (6.3.30), we obtain

$$k_m^{\mathrm{b}}(n)k_m^{\mathrm{f}}(n) = \frac{|\beta_m(n)|^2}{P_m^{\mathrm{f}}(n)P_m^{\mathrm{b}}(n-1)} = \frac{|E\{e_m^{\mathrm{b}}(n-1)e_m^{\mathrm{f}*}(n)\}|^2}{E\{|e_m^{\mathrm{f}}(n)|^2\}E\{|e_m^{\mathrm{b}}(n-1)|^2\}} \tag{6.3.34}$$

which implies that

$$0 \le k_m^{\mathrm{f}}(n)k_m^{\mathrm{b}}(n) \le 1 \tag{6.3.35}$$

because the last term in (6.3.34) is the squared magnitude of the correlation coefficient of the random variables $e_m^{\mathrm{f}}(n)$ and $e_m^{\mathrm{b}}(n-1)$.

Order recursions for the MMSEs. Using the Levinson order recursions, we can obtain order-recursive formulas for the computation of $P_m^{\mathrm{f}}(n)$, $P_m^{\mathrm{b}}(n)$, and $P_m^{\mathrm{c}}(n)$. Indeed, using (6.3.26), (6.3.27), and (6.3.29), we have

$$\begin{aligned}P_{m+1}^{\mathrm{f}}(n) &= P_x(n) + \boldsymbol{r}_{m+1}^{\mathrm{fH}}(n)\boldsymbol{a}_{m+1}(n) \\ &= P_x(n) + [\boldsymbol{r}_m^{\mathrm{fH}}(n) r_{m+1}^{\mathrm{f}*}(n)]\left\{\begin{bmatrix}\boldsymbol{a}_m(n) \\ 0\end{bmatrix} + \begin{bmatrix}\boldsymbol{b}_m(n-1) \\ 1\end{bmatrix} k_m^{\mathrm{f}}(n)\right\} \\ &= P_x(n) + \boldsymbol{r}_m^{\mathrm{fH}}(n)\boldsymbol{a}_m(n) + [\boldsymbol{r}_m^{\mathrm{fH}}(n)\boldsymbol{b}_m(n-1) + r_{m+1}^{\mathrm{f}*}(n)]k_m^{\mathrm{f}}(n)\end{aligned}$$

or

$$P_{m+1}^{\mathrm{f}}(n) = P_m^{\mathrm{f}}(n) + \beta_m^*(n)k_m^{\mathrm{f}}(n) = P_m^{\mathrm{f}}(n) - \frac{|\beta_m(n)|^2}{P_m^{\mathrm{b}}(n-1)} \tag{6.3.36}$$

If we work in a similar manner, we obtain

$$P_{m+1}^{\mathrm{b}}(n) = P_m^{\mathrm{b}}(n-1) + \beta_m(n)k_m^{\mathrm{b}}(n) = P_m^{\mathrm{b}}(n-1) - \frac{|\beta_m(n)|^2}{P_m^{\mathrm{f}}(n)} \tag{6.3.37}$$

and

$$P_{m+1}^{\mathrm{c}}(n) = P_m^{\mathrm{c}}(n) - \beta_m^{\mathrm{c}*}(n)k_m^{\mathrm{c}}(n) = P_m^{\mathrm{c}}(n) - \frac{|\beta_m^{\mathrm{c}}(n)|^2}{P_m^{\mathrm{b}}(n)} \tag{6.3.38}$$

If the subtrahends in the previous recursions are nonzero, increasing the order of the filter always improves the estimates, that is, $\mathrm{P}_{m+1}^{c}(n) \le P_m^c(n)$. Also, the conditions $P_m^f(n) \ne 0$ and $P_m^b(n) \ne 0$ are critical for the invertibility of $\boldsymbol{R}_m(n)$ and the computation of the optimum filters. The above relations are special cases of (6.1.45) The presence of vectors with mixed optimum nesting (upper-lower and lower-upper) in the definitions of $\beta_m(n)$ and $\beta_m^c(n)$ does not lead to similar order recursions for these quantities. However, for stationary processes we can break the dot products in (6.3.17) and (6.3.25) into scalar recursions, using an algorithm first introduced by Schür.

6.3.2 Lattice-Ladder Structure

We saw that the shift invariance of the input data vector made it possible to develop the Levinson recursions for the

BLP and the FLP. We next show that these recursions can be used to simplify the triangular order-recursive estimation structure of Figure 6.1 by reducing it to a more efficient (linear instead of triangular), lattice-ladder filter structure that simultaneously provides the FLP, BLP, and FIR filtering estimates.

The computation of the estimation errors using direct-form structures is based on the following equations:

$$\begin{aligned} e_m^{\mathrm{f}}(n) &= x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n-1) \\ e_m^{\mathrm{b}}(n) &= x(n-m) + \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n) \\ e_m(n) &= y(n) - \boldsymbol{c}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n) \end{aligned} \tag{6.3.39}$$

Using (6.3.2), (6.3.27), and (6.3.39), we obtain

$$\begin{aligned} e_{m+1}^{\mathrm{f}}(n) &= x(n) + \left\{ \begin{bmatrix} \boldsymbol{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \right\}^{\mathrm{H}} \begin{bmatrix} \boldsymbol{x}_m(n-1) \\ x(n-1-m) \end{bmatrix} \\ &= x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n-1) + [\boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{x}_m(n-1) + x(n-1-m)]k_m^{\mathrm{f}*}(n) \end{aligned}$$

or

$$e_{m+1}^{\mathrm{f}}(n) = e_m^{\mathrm{f}}(n) + k_m^{\mathrm{f}*}(n)e_m^{\mathrm{b}}(n-1) \tag{6.3.40}$$

In a similar manner, we obtain

$$e_{m+1}^{\mathrm{b}}(n) = e_m^{\mathrm{b}}(n-1) + k_m^{\mathrm{b}*}(n)e_m^{\mathrm{f}}(n) \tag{6.3.41}$$

Using (6.3.2), (6.3.23), and (6.3.39). Relations (6.3.40) and (6.3.41) are executed for $m = 0, 1, \ldots, M-2$, with $e_0^{\mathrm{f}}(n) = e_0^{\mathrm{b}}(n) = x(n)$, and constitute a lattice filter that implements the FLP and the BLP.

Using (6.3.2), (6.3.15), and (6.3.39), we can show that the optimum filtering error can be computed by

$$e_{m+1}(n) = e_m(n) - k_m^{c*}(n)e_m^b(n) \tag{6.3.42}$$

which is executed for $m = 0, 1, \ldots, M-1$, with $e_0(n) = y(n)$. The last equation provides the ladder part, which is coupled with the lattice predictor to implement the optimum filter. The result is the time-varying lattice-ladder structure shown in Figure 6.3. Notice that a new set of lattice-ladder coefficients has to be computed for every n, using $\boldsymbol{R}_m(n)$ and $\boldsymbol{d}_m(n)$. The parameters of the lattice-ladder structure can be obtained by LDL^H decomposition using (6.3.31). Suppose now that we know $P_0^{\mathrm{f}}(n) = P_0^{\mathrm{b}}(n) = P_x(n)$, $P_0^{\mathrm{b}}(n-1)$, $P_0^{\mathrm{c}}(n) = P_y(n)$, $\{\beta_m(n)\}_0^{M-1}$, and $\{\beta_m^c(n)\}_0^M$. Then we can determine $P_m^f(n), P_m^b(n)$, and $P_m^c(n)$ for all m, using (6.3.16) through (6.3.38), and all filter coefficients, using (6.3.36), (6.3.24), and (6.3.28). However, to obtain a completely time-recursive updating algorithm, we need time updatings for $\beta_m(n)$ and $\beta_m^c(n)$. As we will see later, this is possible if $\boldsymbol{R}(n)$ and $\boldsymbol{d}(n)$ are fixed or are defined by known time-updating formulas.

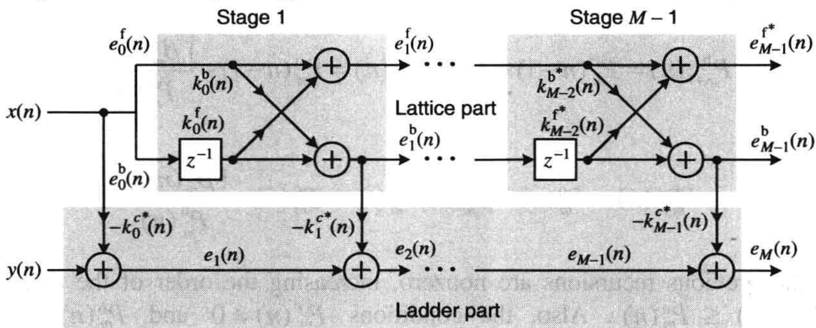


FIGURE 6.3
Lattice-ladder structure for FIR optimum filtering and prediction.

We recall that the BLP error vector $e_{m+1}^{\mathrm{b}}(n)$ is the innovations vector of the data $\mathbf{x}_{m+1}(n)$. Notice that as a result of the shift invariance of the input data vector, the triangular decorrelator of the general linear estimator (see Figure 6.1) is replaced by a simpler, "linear" lattice structure. For stationary processes, the lattice-ladder filter is time-invariant, and we need to compute only one set of coefficients that can be used for all signals with the same $\boldsymbol{R}$

and $\boldsymbol{d}$ (see Section 6.5).

6.3.3 Simplifications for Stationary Stochastic Processes

When $x(n)$ and $y(n)$ are jointly wide-sense stationary (WSS), the optimum estimators are time-invariant and we have the following simplifications:

- All quantities are independent of n; thus we do not need time recursions for the BLP parameters.
- $\boldsymbol{b}_m = \boldsymbol{J}\boldsymbol{a}_m^*$ (see Section 5.5.4), and thus we do not need the Levinson recursion for the BLP $\boldsymbol{b}_m$.

Both simplifications are a consequence of the Toeplitz structure of the correlation matrix $\boldsymbol{R}_m$. Indeed, comparing the partitionings

$$\boldsymbol{R}_{m+1}(n) = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{J}\boldsymbol{r}_m \\ \boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{J} & r(0) \end{bmatrix} = \begin{bmatrix} r(0) & \boldsymbol{r}_m^{\mathrm{T}} \\ \boldsymbol{r}_m^* & \boldsymbol{R}_m \end{bmatrix} \tag{6.3.43}$$

where

$$r_m \triangleq [r(1)\ r(2)\ \cdots\ r(m)]^{\mathrm{T}} \tag{6.3.44}$$

with (6.3.3) and (6.3.4), we have

$$\begin{aligned} \boldsymbol{R}_m(n) &= \boldsymbol{R}_m(n-1) = \boldsymbol{R}_m \\ \boldsymbol{r}_m^{\mathrm{f}}(n) &= \boldsymbol{r}_m^* \\ \boldsymbol{r}_m^{\mathrm{b}}(n) &= \boldsymbol{J}\boldsymbol{r}_m \end{aligned} \tag{6.3.45}$$

which can be used to simplify the order recursions derived for nonstationary processes. Indeed, we can easily show that

$$\boldsymbol{a}_{m+1} = \begin{bmatrix} \boldsymbol{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix} k_m \tag{6.3.46}$$

where

$$\boldsymbol{b}_m = \boldsymbol{J}\boldsymbol{a}_m^* \tag{6.3.47}$$

$$k_m \triangleq k_m^{\mathrm{f}} = k_m^{\mathrm{b}*} = -\frac{\beta_m}{P_m} \tag{6.3.48}$$

$$\beta_m \triangleq \beta_m^{\mathrm{f}} = \beta_m^{\mathrm{b}*} = \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{r}_m^* + r^*(m+1) \tag{6.3.49}$$

$$P_m \triangleq P_m^{\mathrm{b}} = P_m^{\mathrm{f}} = P_{m-1} + \beta_{m-1}^* k_{m-1} = P_{m-1} + \beta_{m-1} k_{m-1}^* \tag{6.3.50}$$

This recursion provides a complete order-recursive algorithm for the computation of the FLP $\mathbf{a}_m$ for $1 \le m \le M$ from the autocorrelation sequence $r(l)$ for $0 \le l \le M$.

The optimum filters $\boldsymbol{c}_m$ for $1 \le m \le M$ can be obtained from the quantities $\boldsymbol{a}_m$ and P_m for $1 \le m \le M-1$ and $\boldsymbol{d}_M$, using the following Levinson recursion

$$\boldsymbol{c}_{m+1} = \begin{bmatrix} \boldsymbol{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_m \\ 1 \end{bmatrix} k_m^c \tag{6.3.51}$$

where

$$k_m^c \triangleq \frac{\beta_m^c}{P_m} \tag{6.3.52}$$

and

$$\beta_m^c = \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{d}_m + d_{m+1} \tag{6.3.53}$$

The MMSE P_m^c is then given by

$$P_m^c = P_{m-1}^c - \beta_m^c k_m^c \tag{6.3.54}$$

and although it is not required by the algorithm, P_m^c is useful for selecting the order of the optimum filter. Both algorithms are discussed in greater detail in Section 6.4.

6.4 Algorithms of Levinson and Levinson-Durbin

Since the correlation matrix of a stationary, stochastic process is Toeplitz, we can explore its special structure to develop efficient, order-recursive algorithms for the linear system solution, matrix triangularization, and matrix inversion. Although we develop such algorithms in the context of optimum FIR filtering and prediction, the results apply to other applications involving Toeplitz matrices (Golub and van Loan 1996).

Suppose that we know the optimum filter $\boldsymbol{c}_m$ is given by

$$\boldsymbol{c}_m = \boldsymbol{R}_m^{-1}\boldsymbol{d}_m \tag{6.4.1}$$

and we wish to use it to compute the optimum filter $\boldsymbol{c}_{m+1}$

$$\boldsymbol{c}_{m+1} = \boldsymbol{R}_{m+1}^{-1}\boldsymbol{d}_{m+1} \tag{6.4.2}$$

We first notice that the matrix $\boldsymbol{R}_{m+1}$ and the vector $\boldsymbol{d}_{m+1}$ can be partitioned as follows

$$\mathbf{R}_{m+1} = \left[\begin{array}{ccc|c} r(0) & \cdots & r(m-1) & r(m) \\ \vdots & \ddots & \vdots & \vdots \\ r^*(m-1) & \cdots & r(0) & r(1) \\ \hline r^*(m) & \cdots & r^*(1) & r(0) \end{array}\right] = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{J}\boldsymbol{r}_m \\ \boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{J} & r(0) \end{bmatrix} \tag{6.4.3}$$

$$\boldsymbol{d}_{m+1} = \begin{bmatrix} \boldsymbol{d}_m \\ d_{m+1} \end{bmatrix} \tag{6.4.4}$$

which shows that both quantities have the optimum nesting property, that is, $\boldsymbol{R}_{m+1}^{[m]} = \boldsymbol{R}_m$ and $\boldsymbol{d}_{m+1}^{[m]} = \boldsymbol{d}_m$.

Using the matrix inversion by partitioning lemma (6.1.24), we obtain

$$\boldsymbol{R}_{m+1}^{-1} = \begin{bmatrix} \boldsymbol{R}_m^{-1} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{H}} & 0 \end{bmatrix} + \frac{1}{P_m^{\mathrm{b}}}\begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{b}_m^{\mathrm{H}} & 1 \end{bmatrix} \tag{6.4.5}$$

where

$$\boldsymbol{b}_m = -\boldsymbol{R}_m^{-1}\boldsymbol{J}\boldsymbol{r}_m \tag{6.4.6}$$

and

$$P_m^{\mathrm{b}} = r(0) + \boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{J}\boldsymbol{b}_m \tag{6.4.7}$$

Substitution of (6.4.4) and (6.4.5) into (6.4.2) gives

$$\boldsymbol{c}_{m+1} = \begin{bmatrix} \boldsymbol{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix} k_m^c \tag{6.4.8}$$

where

$$k_m^c \triangleq \frac{\beta_m^c}{P_m^{\mathrm{b}}} \tag{6.4.9}$$

and

$$\beta_m^c \triangleq \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{d}_m + d_{m+1} = -\boldsymbol{c}_m^{\mathrm{H}}\boldsymbol{J}\boldsymbol{r}_m + d_{m+1} \tag{6.4.10}$$

Equations (6.4.8) through (6.4.10) constitute a Levinson recursion for the optimum filter and have been obtained

without making use of the Toeplitz structure of $\boldsymbol{R}_{m+1}$.

The development of a complete order-recursive algorithm is made possible by exploiting the Toeplitz structure. Indeed, when the correlation matrix $\boldsymbol{R}_m$ is Toeplitz, we have

$$\boldsymbol{b}_m = \boldsymbol{J}\boldsymbol{a}_m^* \tag{6.4.11}$$

and

$$P_m \triangleq P_m^b = P_m^f \tag{6.4.12}$$

as we recall from Section 5.5. Since we an determine $\boldsymbol{b}_m$ from $\mathbf{a}_m$, we need to perform only *one* Levinson recursion, either for $\boldsymbol{b}_m$ or for $\boldsymbol{a}_m$.

To avoid the use of the lower right corner partitioning, we develop an order recursion for the FLP $\boldsymbol{a}_m$. Indeed, to compute $\boldsymbol{a}_{m+1}$ from $\boldsymbol{a}_m$, recall that

$$\boldsymbol{a}_{m+1} = -\boldsymbol{R}_{m+1}^{-1}\boldsymbol{r}_{m+1}^* \tag{6.4.13}$$

which, when combined with (6.4.5) and

$$\boldsymbol{r}_{m+1} = \begin{bmatrix} \boldsymbol{r}_m \\ r(m+1) \end{bmatrix} \tag{6.4.14}$$

leads to the Levinson recursion

$$\boldsymbol{a}_{m+1} = \begin{bmatrix} \boldsymbol{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix} k_m \tag{6.4.15}$$

where

$$k_m \triangleq -\frac{\beta_m}{P_m} \tag{6.4.16}$$

$$\beta_m \triangleq \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{r}_m^* + r^*(m+1) = \boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{J}\boldsymbol{r}_m^* + r^*(m+1) \tag{6.4.17}$$

and

$$P_m = r(0) + \boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{a}_m^* = r(0) + \boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{r}_m \tag{6.4.18}$$

Also, using (6.1.46) and (6.2.6), we can show that

$$\det \boldsymbol{R}_m = \prod_{i=0}^{m-1} P_i \qquad \text{with } P_0 = r(0) \tag{6.4.19}$$

which emphasizes the importance of P_m for the invertibility of the autocorrelation matrix. The MMSE P_m for either the forward or the backward predictor of order m can be computed recursively as follows:

$$\begin{aligned} P_{m+1} &= r(0) + [\boldsymbol{r}_m^{\mathrm{H}} \; r^*(m+1)]\left\{\begin{bmatrix} \boldsymbol{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_m \\ 1 \end{bmatrix} k_m\right\}^* \\ &= r(0) + \boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{a}_m^* + [\boldsymbol{r}_m^{\mathrm{H}}\boldsymbol{b}_m^* + r^*(m+1)]k_m^* \end{aligned} \tag{6.4.20}$$

or

$$P_{m+1} = P_m + \beta_m k_m^* = P_m + \beta_m^* k_m \tag{6.4.21}$$

The following recrusive formula for the computation of the MMSE

$$P_{m+1}^c = P_m^c - \beta_m^c k_m^{c*} = P_m^c - \beta_m^{c*} k_m^c \tag{6.4.22}$$

can be found by using (6.4.8).

Therefore, the algorithm of Levinson consists of two parts: a set of recursions that compute the optimum FLP or BLP and a set of recursions that use this information to compute the optimum filter. The part that computes the linear predictors is known as the Levinson-Durbin algorithm and was pointed out by Durbin (1960). From a linear system

solution point of view, the algorithm of Levinson solves a Hermitian Toeplitz system with *arbitrary* right-hand side vector $\boldsymbol{d}$; the Levinson-Durbin algorithm deals with the special case $\boldsymbol{d} = \boldsymbol{r}^*$ or $\boldsymbol{Jr}$.

Algorithm of Levinson-Durbin

The algorithm of Levinson-Durbin, which takes as input the autocorrelation sequence $r(0), r(1), \cdots, r(M)$ and computes the quantities $\mathbf{a}_m$, P_m, and k_{m-1} for $m = 1, 2, \cdots, M$, is illustrated in the following examples.

EXMPLE 6.4.1. Determine the FLP $\boldsymbol{a}_2 = [a_1^{(2)} \quad a_2^{(2)}]^T$ and the MMSE P_2 from the autocorrelation values $r(0), r(1)$, and $r(2)$.

Solution. To initialize the algorithm, we determine the first-order predictor by solving the normal equations $r(0)a_1^{(1)} = -r^*(1)$. Indeed, we have

$$a_1^{(1)} = -\frac{r^*(1)}{r(0)} = k_0 = -\frac{\beta_0}{P_0}$$

which implies that $\qquad \beta_0 = r^*(1) \qquad P_0 = r(0)$

To update to order 2, we need k_1 and hence β_1 and P_1, which can be obtained by

$$\beta_1 = a_1^{(1)} r^*(1) + r^*(2) = \frac{r(0)r^*(2) - [r^*(1)]^2}{r(0)}$$

$$P_1 = P_0 + \beta_0 k_0^* = \frac{r^2(0) - |r(1)|^2}{r(0)}$$

as

$$k_1 = \frac{[r^*(1)]^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}$$

Therefore, using Levinson's recursion, we obtain

$$a_1^{(2)} = a_1^{(1)} + a_1^{(1)*} k_1 = \frac{[r(1)r^*(2)]^2 - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

and

$$a_2^{(2)} = k_1$$

which agree with the results obtained in Example 5.5.1. The resulting MMSE can be found by using $P_2 = P_1 + \beta_1 k_1^*$.

EXAMPLE 6.4.2 Use the Levinson-Durbin algorithm to compute the third-order forward predictor for a signal $x(n)$ with autocorrelation sequence $r(0) = 3$, $r(1) = 2$, $r(2) = 1$, and $r(3) = \frac{1}{2}$.

Solution. To initialize the algorithm, we notice that the first-order predictor is given by $r(0)a_1^{(1)} = -r(1)$ and that for $m = 0$, (6.4.15) gives $a_1^{(1)} = k_0$. Hence, we have

$$a_1^{(1)} = -\frac{r(1)}{r(0)} = -\frac{2}{3} = \frac{k_0}{k_1} = \frac{\beta_0}{P_0}$$

which implies $\qquad P_0 = r(0) = 3 \qquad \beta_0 = r(1) = 2$

To compute $\mathbf{a}_2$ by (6.4.15), we need $a_1^{(1)}$, $b_1^{(1)} = a_1^{(1)}$, and $k_1 = -\beta_1 / P_1$ From (6.4.21), we have

$$P_1 = P_0 + \beta_0 k_0 = 3 + 2(-\frac{2}{3}) = \frac{5}{3}$$

and from (6.4.17)

$$\beta_1 = \mathbf{r}_1^T \mathbf{J} \mathbf{a}_1 + r(2) = 2(-\frac{2}{3}) + 1 = -\frac{1}{3}$$

Hence,

$$k_1 = -\frac{\beta_1}{P_1} = -\frac{-\frac{1}{3}}{\frac{5}{3}} = \frac{1}{5}$$

and

$$\mathbf{a}_2 = \begin{bmatrix} -\frac{2}{3} \\ 0 \end{bmatrix} + \frac{1}{5}\begin{bmatrix} -\frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{4}{5} \\ \frac{1}{5} \end{bmatrix}$$

Continuing in the same manner, we obtain

$$P_2 = P_1 + \beta_1 k_1 = \frac{5}{3} + (-\frac{1}{3})(\frac{1}{5}) = \frac{8}{5}$$

$$\beta_2 = \boldsymbol{r}_2^{\mathrm{T}} \boldsymbol{J}\boldsymbol{a}_2 + r(3) = [2 \quad 1]\begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \end{bmatrix} + \frac{1}{2} = \frac{1}{10}$$

$$k_2 = -\frac{\beta_2}{P_2} = -\frac{\frac{1}{10}}{\frac{8}{5}} = -\frac{1}{16}$$

$$\boldsymbol{a}_3 = \begin{bmatrix} \boldsymbol{a}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_2 \\ 1 \end{bmatrix} k_3 = \begin{bmatrix} -\frac{4}{5} \\ \frac{1}{5} \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \\ 1 \end{bmatrix} \frac{1}{16} = \begin{bmatrix} -\frac{13}{16} \\ \frac{1}{4} \\ -\frac{1}{16} \end{bmatrix}$$

$$P_3 = P_2 + \beta_2 k_2 = \frac{8}{5} + \frac{1}{10}\left(-\frac{1}{16}\right) = \frac{51}{32}$$

The algorithm of Levinson-Durbin, summarized in Table 6.2, requires M^2 operations and is implemented by the function `[a,k,Po]=durbin(r,M)`.

TABLE 6.2.
Summary of the Levinson-Durbin algorithm.

1. **Input:** $r(0), r(1), r(2), \cdots, r(M)$
2. **Initialization**
   (a) $P_0 = r(0)$, $\beta_0 = r^*(1)$
   (b) $k_0 = -r^*(1)/r(0)$, $a_1^{(1)} = k_0$
3. **For** $m = 1, 2, \cdots, M-1$
   (a) $P_m = P_{m-1} + \beta_{m-1} k_{m-1}^*$
   (b) $\boldsymbol{r}_m = [r(1)\ r(2)\ \cdots\ r(m)]^{\mathrm{T}}$
   (c) $\beta_m = \boldsymbol{a}_m^{\mathrm{T}} \boldsymbol{J} \boldsymbol{r}_m^* + r^*(m+1)$
   (d) $k_m = -\beta_m / P_m$
   (e) $\boldsymbol{a}_{m+1} = \begin{bmatrix} \boldsymbol{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_m^* \\ 1 \end{bmatrix} k_m$
4. $P_M = P_{M-1} + \beta_M k_M^*$
5. **Output:** $\boldsymbol{a}_M, \{k_m\}_0^{M-1}, \{P_m\}_1^M$

Algorithm of Levinson

The next example illustrates the algorithm of Levinson that can be used to solve a system of linear equations with a Hermitian Toeplitz matrix and arbitrary right-hand side vector.

EXAMPLE 6.4.3 Consider an optimum filter with input $x(n)$ and desired response $y(n)$. The autocorrelation of the input signal is $r(0)=3$, $r(1)=2$, and $r(2)=1$. The cross-correlation between the desired response and input is $d_1=1$, $d_2=2$, and $d_3=\frac{5}{2}$; and the power of $y(n)$ is $P_y=3$. Design a third-order optimum FIR filter, using the algorithm of Levinson.

Solution. We start initializing the algorithm by noticing that for $m=0$ we have $r(0)a_1^{(1)}=-r(1)$, which gives

$$a_1^{(1)}=k_0=-\frac{r(1)}{r(0)}=-\frac{2}{3}$$

$$P_0=r(0)=3 \qquad \beta_0=r(1)=2$$

and

$$P_1=P_0+\beta_0 k_0=3+2(-\frac{2}{3})=\frac{5}{3}$$

Next, we compute the Levinson recursion for the first-order optimum filter

$$P_0^c=5 \qquad \beta_0^c=d_1=1$$

$$k_0^c=c_1^{(1)}=\frac{d_1}{r(0)}=\frac{1}{3}$$

$$P_1^c=P_0^c-\beta_0^c k_0^c=3-1(\frac{1}{3})=\frac{8}{3}$$

Then we carry the Levinson recursion for $m=1$ to obtain

$$\beta_1=\boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{J}\boldsymbol{a}_1+r(2)=2(-\frac{2}{3})+1=-\frac{1}{3}$$

$$k_1=-\frac{\beta_1}{P_1}=-\frac{-\frac{1}{3}}{\frac{5}{3}}=\frac{1}{5}$$

$$\boldsymbol{a}_2=\begin{bmatrix}-\frac{2}{3}\\0\end{bmatrix}+\frac{1}{5}\begin{bmatrix}-\frac{2}{3}\\1\end{bmatrix}=\begin{bmatrix}-\frac{4}{5}\\\frac{1}{5}\end{bmatrix}$$

$$P_2=P_1+\beta_1 k_1=\frac{5}{3}+(-\frac{1}{3})(\frac{1}{5})=\frac{8}{5}$$

for the optimum predictor, and

$$\beta_1^c=\boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{J}\boldsymbol{d}_1+d_2=-\frac{2}{3}(1)+2=\frac{4}{3}$$

$$k_1^c=\frac{\beta_1^c}{P_1}=\frac{\frac{4}{3}}{\frac{5}{3}}=\frac{4}{5}$$

$$\boldsymbol{c}_2=\begin{bmatrix}\frac{1}{3}\\0\end{bmatrix}+\frac{4}{5}\begin{bmatrix}-\frac{2}{3}\\1\end{bmatrix}=\begin{bmatrix}-\frac{1}{5}\\\frac{4}{5}\end{bmatrix}$$

$$P_2^c=P_1^c-\beta_1^c k_1^c=\frac{8}{3}-\frac{4}{3}(\frac{4}{5})=\frac{8}{5}$$

for the optimum filter. The last recursion $(m=2)$ is carried out only for the optimum filter and gives

$$\beta_2^c = \boldsymbol{a}_2^{\mathrm{T}} \boldsymbol{J} \boldsymbol{d}_2 + d_3 = \begin{bmatrix} \frac{1}{5} & -\frac{4}{5} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \frac{5}{2} = \frac{11}{10}$$

$$k_2^c = \frac{\beta_2^c}{P_2} = \frac{\frac{11}{10}}{\frac{8}{5}} = \frac{11}{16}$$

$$\boldsymbol{c}_3 = \begin{bmatrix} \boldsymbol{c}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_2 \\ 1 \end{bmatrix} k_2^c = \begin{bmatrix} -\frac{1}{5} \\ \frac{4}{5} \\ 0 \end{bmatrix} + \frac{11}{16} \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{16} \\ \frac{1}{4} \\ \frac{11}{16} \end{bmatrix}$$

$$P_3^c = P_2^c - \beta_2^c k_2^c = \frac{8}{5} - \frac{11}{10}(\frac{11}{16}) = \frac{27}{32}$$

The algorithm of Levinson, summarized in Table 6.3, is implemented by the MATLAB function `[c,k,kc,Pc]=levins(R,d,Py,M)` and requires $2M^2$ operations because it involves two dot products and two scalar-vector multiplications. A parallel processing implementation of the algorithm is not possible because the dot products involve additions that cannot be executed simultaneously. Notice that adding $M = 2^q$ numbers using $M/2$ adders requires $q = \log_2 M$ steps.

Minimum phase and autocorrelation extension

Using (6.4.16), we can also express the recursion (6.4.21) as

$$P_{m+1} = P_m(1 - |k_m|^2) = P_m - \frac{|\beta_m|^2}{P_m} \tag{6.4.23}$$

TABLE 6.3.
Summary of the algorithm of Levinson.

1 **Input:** $\{r(l)\}_0^M, \{d_m\}_1^M, P_y$

2 **Initialization**

  (a) $P_0 = r(0),\ \beta_0 = r^*(1),\ P_0^c = P_y$

  (b) $k_0 = -\beta_0/P_0,\ a_1^{(1)} = k_0$

  (c) $\beta_0^c = d_1$

  (d) $k_0^c = -\beta_0^c/P_0,\ c_1^{(1)} = k_0^c$

  (e) $P_1^c = P_0^c + \beta_0^c k_0^{c*}$

3 **For** $m = 1, 2, \cdots, M-1$

  (a) $\boldsymbol{r}_m = [r(1)\ r(2)\ \cdots\ r(m)]^T$

  (b) $\beta_m = \boldsymbol{a}_m^{\mathrm{T}} \boldsymbol{J} \boldsymbol{r}_m^* + r^*(m+1)$

  (c) $P_m = P_{m-1} + \beta_{m-1} k_{m-1}^*$

  (d) $k_m = -\beta_m / P_m$

  (e) $\boldsymbol{a}_{m+1} = \begin{bmatrix} \boldsymbol{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_m^* \\ 1 \end{bmatrix} k_m$

  (f) $\beta_m^c = -\boldsymbol{c}_m^{\mathrm{H}} \boldsymbol{J} \boldsymbol{r}_m + d_{m+1}$

  (g) $k_m^c = \beta_m^c / P_m$

  (h) $\boldsymbol{c}_{m+1} = \begin{bmatrix} \boldsymbol{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}_m^* \\ 1 \end{bmatrix} k_m^c$

  (i) $P_{m+1}^c = P_m^c + \beta_m^c k_m^{c*}$

4 **Output:** $\boldsymbol{a}_M,\ \boldsymbol{c}_M,\ \{k_m,\ k_m^c\}_0^{M-1},\ \{P_m,\ P_m^c\}_0^M$

which, since $P_m \geq 0$, implies that

$$P_{m+1} \leq P_m \tag{6.4.24}$$

and since the matrix $\mathbf{R}_m$ is positive definite, then $P_m > 0$ and (6.4.23) implies that

$$|k_m| \leq 1 \tag{6.4.25}$$

for all $1 \leq m < M$. If

$$P_0 > \cdots > P_{M-1} > P_M = 0 \tag{6.4.26}$$

then the process $x(n)$ is predictable and (6.4.23) implies that

$$k_M = \pm 1 \qquad \text{and} \qquad |k_m| < 1 \qquad 1 \leq k < M \tag{6.4.27}$$

(see Section 5.6.4). Also if

$$P_{M-1} > P_M = \cdots = P_\infty = P > 0 \tag{6.4.28}$$

from (6.4.23) we have

$$k_m = 0 \qquad \text{for } m > M \tag{6.4.29}$$

which implies that the process $x(n)$ is AR(M) and $e_M^{\mathrm{f}}(n) \sim \mathrm{WN}(0, P_M)$ (see Section 3.2.3). Finally, we note that since the sequence P_0, P_1, P_2, ... is nonincreasing, its limit as $m \to \infty$ exists and is nonnegative. A regular process must satisfy $|k_m| < 1$ for all m, because $|k_m| = 1$ implies that $P_m = 0$, which contradicts the regularity assumption.

For $m = 0$, (6.4.19) gives $P_0 = r(0)$. Carrying out (6.4.23) from $m = 0$ to $m = M$, we obtain

$$P_M = r(0) \prod_{m=1}^{M} (1 - |k_{m-1}|^2) \tag{6.4.30}$$

which converges, as $M \to \infty$, if $|k_m| < 1$.

6.5 Lattice Structures for Optimum Fir Filters And Predictors

To compute the forward prediction error of an FLP of order m, we use the formula

$$e_m^{\mathrm{f}}(n) = x(n) + \mathbf{a}_m^{\mathrm{H}} \mathbf{x}_m(n-1) = x(n) + \sum_{k=1}^{m} a_k^{(m)*} x(n-k) \tag{6.5.1}$$

Similarly, for the BLP we have

$$e_m^{\mathrm{b}}(n) = x(n-m) + \mathbf{b}_m^{\mathrm{H}} \mathbf{x}_m(n) = x(n-m) + \sum_{k=0}^{m-1} b_k^{(m)*} x(n+1-k) \tag{6.5.2}$$

Both filters can be implemented using the direct-form filter structure shown in Figure 6.4. Since $\mathbf{a}_m$ and $\mathbf{b}_m$ do not have the optimum nesting property, we cannot obtain order-recursive direct-form structures for the computation of the prediction errors. However, next we show that we can derive an order-recursive lattice-ladder structure for the implementation of optimum predictors and filters using the algorithm of Levinson.

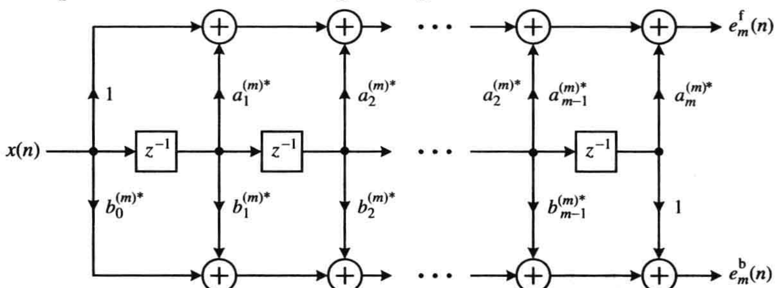


FIGURE 6.4

Direct-form structure for the computation of the m th-order forward and backward prediction errors.

6.5.1 Lattice-Ladder Structures

We note that the data vector for the $(m+1)$ st-order predictor can be partitioned in the following ways:

$$\boldsymbol{x}_{m+1}(n) = [x(n)x(n-1)\cdots x(n-m+1)x(n-m)]^{\mathrm{T}}$$

$$=[\boldsymbol{x}_m^{\mathrm{T}}(n)x(n-m)]^{\mathrm{T}} \tag{6.5.3}$$

$$=[x(n)\boldsymbol{x}_m^{\mathrm{T}}(n-1)]^{\mathrm{T}} \tag{6.5.4}$$

Using (6.5.1), (6.5.3), (6.4.15), and (6.5.2), we obtain

$$e_{m+1}^{f}(n) = x(n) + \left\{\begin{bmatrix}\boldsymbol{a}_m\\0\end{bmatrix} + \begin{bmatrix}\boldsymbol{b}_m\\1\end{bmatrix}k_m\right\}^{\mathrm{H}}\begin{bmatrix}\boldsymbol{x}_m(n-1)\\x(n-m-1)\end{bmatrix}$$

$$= x(n) + \boldsymbol{a}_m^{\mathrm{H}}\boldsymbol{x}_m(n-1) + k_m^*[\boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{x}_m(n-1) + x(n-1-m)]$$

or

$$e_{m+1}^{\mathrm{f}}(n) = e_m^{\mathrm{f}}(n) + k_m^* e_m^{\mathrm{b}}(n-1) \tag{6.5.5}$$

Using (6.4.11) and (6.4.15), we obtain the following Levinson-type recursion for the backward predictor:

$$\boldsymbol{b}_{m+1} = \begin{bmatrix}0\\\boldsymbol{b}_m\end{bmatrix} + \begin{bmatrix}1\\\boldsymbol{a}_m\end{bmatrix}k_m^*$$

The backward prediction error is

$$e_{m+1}^{\mathrm{b}}(n) = x(n-m-1) + \left\{\begin{bmatrix}0\\\boldsymbol{b}_m\end{bmatrix} + \begin{bmatrix}1\\\boldsymbol{a}_m\end{bmatrix}k_m^*\right\}^{\mathrm{H}}\begin{bmatrix}x(n)\\\boldsymbol{x}_m(n-1)\end{bmatrix}$$

$$= x(n-m-1) + \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{x}_m(n-1) + k_m[x(n) + \boldsymbol{a}_m^{\mathrm{H}}\boldsymbol{x}_m(n-1)]$$

or

$$e_{m+1}^{b}(n) = e_m^{b}(n-1) + k_m e_m^{f}(n) \tag{6.5.6}$$

Recursions (6.5.5) and (6.5.6) can be computed for $m = 0, 1, \cdots, M-1$. The initial conditions $e_0^f(n)$ and $e_0^b(n)$ are easily obtained from (6.5.1) and (6.5.2). The recursions also lead to the following all-zero lattice algorithm

$$\begin{aligned}
&e_0^{\mathrm{f}}(n) = e_0^{\mathrm{b}}(n) = x(n)\\
&e_m^{\mathrm{f}}(n) = e_{m-1}^{\mathrm{f}}(n) + k_{m-1}^* e_{m-1}^{\mathrm{b}}(n-1) \qquad m = 1,\ 2,\ \cdots,\ M\\
&e_m^{\mathrm{b}}(n) = k_{m-1} e_{m-1}^{\mathrm{f}}(n) + e_{m-1}^{\mathrm{b}}(n-1) \qquad m = 1,\ 2,\ \cdots,\ M\\
&e(n) = e_M^{\mathrm{f}}(n)
\end{aligned} \tag{6.5.7}$$

that is implemented using the structure shown in Figure 6.5. The *lattice parameters* k_m are known as *reflection coefficients* in the speech processing and geophysics areas.

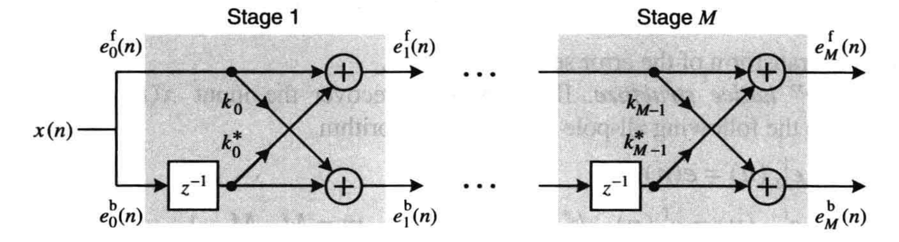


FIGURE 6.5
All-zero lattice structure for the implementation of the forward and backward prediction error filters.

The Levinson recursion for the optimum filter, (6.4.8) through (6.4.10), adds a *ladder* part to the lattice structure for the forward and backward predictors. Using (6.4.8), (6.5.7), and the partitioning in (6.5.3), we can express the

filtering error of order $m+1$ in terms of $e_m(n)$ and $e_m^b(n)$ as follows

$$e_{m+1}(n) = y(n) - \boldsymbol{c}_{m+1}^{\mathrm{H}} \boldsymbol{x}_{m+1}(n) = e_m(n) - k_m^{c*} e_m^{\mathrm{b}}(n) \tag{6.5.8}$$

for $m = 0, 1, \cdots, M-1$. The resulting lattice-ladder structure is similar to the one shown in Figure 6.3. However, owing to stationarity all coefficients are constant, and $k_m^f(n) = k_m^b(n) = k_m$. We note that the efficient solution of the Mth-order optimum filtering problem is derived from the solution of the $(M-1)$ st-order forward and backward prediction problems of the input process. In fact, the lattice part serves to decorrelate the samples $x(n), x(n-1), \cdots, x(n-M)$, producing the uncorrelated samples $e_0^b(n), e_1^b(n), \cdots, e_M^b(n)$ (innovations), which are then linearly combined ("recorrelated") to obtain the optimum estimate of the desired response.

System functions. We next express the various lattice relations in terms of z-transforms. Taking the z-transform of (6.5.1) and (6.5.2), we obtain

$$E_m^{\mathrm{f}}(z) = \left(1 + \sum_{k=1}^{M} a_k^{(m)*} z^{-k}\right) X(z) \triangleq A_m(z) X(z) \tag{6.5.9}$$

$$E_m^{\mathrm{b}}(z) = \left(z^{-m} + \sum_{k=1}^{M} b_k^{(m)*} z^{-k+1}\right) X(z) \triangleq B_m(z) X(z) \tag{6.5.10}$$

where $A_m(z)$ and $B_m(z)$ are the system functions of the paths from the input to the outputs of the m th stage of the lattice. Using the symmetry relation $\boldsymbol{a}_m = \boldsymbol{J}\boldsymbol{b}_m^*$, $1 \le m \le M$, we obtain

$$B_m(z) = z^{-m} A_m^*\left(\frac{1}{z^*}\right) \tag{6.5.11}$$

Note that if z_0 is a zero of $A_m(z)$, then z_0^{-1} is a zero of $B_m(z)$. Therefore, if $A_m(z)$ is minimum-phase, then $B_m(z)$ is maximum-phase.

Taking the z-transform of the lattice equations (6.5.7), we have for the mth stage

$$E_m^{\mathrm{f}}(z) = E_{m-1}^{\mathrm{f}}(z) + k_{m-1}^* z^{-1} E_{m-1}^{\mathrm{b}}(z) \tag{6.5.12}$$

$$E_m^{\mathrm{b}}(z) = k_{m-1} E_{m-1}^{\mathrm{f}}(z) + z^{-1} E_{m-1}^{\mathrm{b}}(z) \tag{6.5.13}$$

Dividing both equations by $X(z)$ and using (6.5.9) and (6.5.10), we have

$$A_m(z) = A_{m-1}(z) + k_{m-1}^* z^{-1} B_{m-1}(z) \tag{6.5.14}$$

$$B_m(z) = k_{m-1} A_{m-1}(z) + z^{-1} B_{m-1}(z) \tag{6.5.15}$$

which, when initialized with

$$A_0(z) = B_0(z) = 1 \tag{6.5.16}$$

describe the lattice filter in the z domain.

The z-transform of the ladder-part (6.5.8) is given by

$$E_{m+1}(z) = E_m(z) - k_m^{c*} E_m^b(z) \tag{6.5.17}$$

where $E_m(z)$ is the z-transform of the error sequence $e_m(n)$.

All-pole or "inverse" lattice structure. If we wish to recover the input $x(n)$ from the prediction error $e(n) = e_M^f(n)$, we can use the following all-pole lattice filter algorithm

$$\begin{aligned} & e_M^{\mathrm{f}}(n) = e(n) \\ & e_{m-1}^{\mathrm{f}}(n) = e_m^{\mathrm{f}}(n) - k_{m-1}^* e_{m-1}^{\mathrm{b}}(n-1) \qquad m = M, M-1, \cdots, 1 \\ & e_m^{\mathrm{b}}(n) = e_{m-1}^{\mathrm{b}}(n-1) + k_{m-1} e_{m-1}^{\mathrm{f}}(n) \qquad m = M, M-1, \cdots, 1 \\ & x(n) = e_0^{\mathrm{f}}(n) = e_0^{\mathrm{b}}(n) \end{aligned} \tag{6.5.18}$$

which is implemented by using the structure in Figure 6.6.

Although the system functions of the all-zero lattice in (6.5.7) and the all-pole lattice in (6.5.18) are

$H_{AZ}(z)=A(z)$ and $H_{AP}(z)=1/A(z)$, the two lattice structures are described by the same set of lattice coefficients. The difference is the signal flow (see feedback loops in the all-pole structure). This structure is used in speech processing applications (Rabiner and Schafer 1978).

6.5.2 Some Properties and Interpretations

Lattice filters have some important properties and interesting interpretations that make them a useful tool in optimum filtering and signal modeling.

Optimal nesting. The all-zero lattice filter has an *optimal nesting* property when it is used for the implementation of an FLP. Indeed, if we use the lattice parameters obtained via the algorithm of Levinson-Durbin, the all-zero lattice filter driven by the signal $x(n)$ produces prediction errors $e_m^f(n)$ and $e_m^b(n)$ at the output of the m th stage for all $1 \le m \le M$. This implies that we can increase the order of the filter by attaching additional stages without destroying the optimality of the previous stages. In contrast, the direct-form filter structure implementation requires the computation of the entire predictor for each stage. However, the nesting property does not hold for the all-pole lattice filter because of the feedback path.

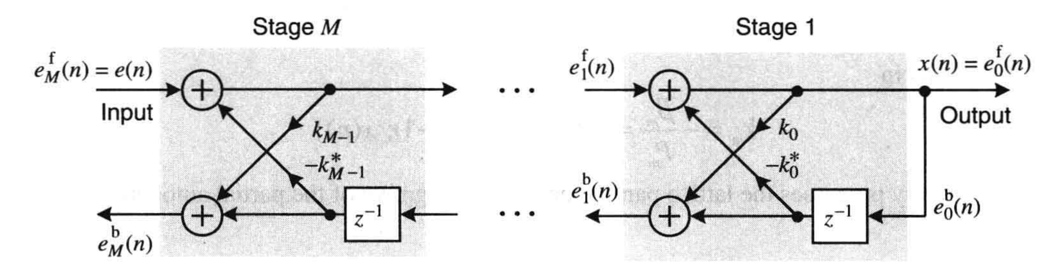


FIGURE 6.5

All-pole lattice structure for recovering the input signal from the forward prediction error.

Orthogonality. The backward prediction errors $e_m^b(n)$ for $0 \le m \le M$ are uncorrelated (see Section 6.2), that is,

$$E\{e_m^b(n)e_k^{b*}(n)\} = \begin{cases} P_m & k=m \\ 0 & k \ne m \end{cases} \tag{6.5.19}$$

and constitute the innovations representation of the input samples $x(n), x(n-1), \cdots, x(n-m)$. We see that at a given time instant n, the backward prediction errors for orders $m = 0, 1, 2, \cdots, M$ are uncorrelated and are part of a nonstationary sequence because the variance $E|e_m^b(n)|^2\} = P_m$ depends on n. This should be expected because, for a given n, each $e_m^b(n)$ is computed using a different set of predictor coefficients. In contrast, for a given m, the sequence $e_m^b(n)$ is stationary for $-\infty < n < \infty$.

Reflection coefficients. The all-pole lattice structure is very useful in the modeling of layered media, where each stage of the lattice models one layer or section of the medium. Traveling waves in geophysical layers, in acoustic tubes of varying cross-sections, and in multisectional transmission lines have been modeled in this fashion. The modeling is performed such that the wave travel time through each section is the same, but the sections may have different impedances. The m th section is modeled with the signals $e_m^f(n)$ and $e_m^b(n)$ representing the forward and backward traveling waves, respectively.

If Z_m and Z_{m-1} are the characteristic impedances at sections m and $m-1$, respectively, then k_m represents the reflection coefficients between the two sections, given by

$$k_m = \frac{Z_m - Z_{m-1}}{Z_m + Z_{m-1}} \tag{6.5.20}$$

For this reason, the lattice parameters k_m are often known as *reflection coefficients*. As reflection coefficients, it makes good sense that their magnitudes not exceed unity. The termination of the lattice assumes a perfect reflection, and so the reflected wave $e_0^b(n)$ is equal to the transmitted wave $e_0^f(n)$. The result of this specific termination is an overall all-pole model (Rabiner and Schafer 1978).

Partial correlation coefficients. The *partial correlation coefficient* (*PCC*) between $x(n)$ and $x(n-m-1)$ (see also Section 6.2.2) is defined as the correlation coefficient between $e_m^f(n)$ and $e_m^b(n-1)$, that is,

$$\mathrm{PCC}\{x(n-m-1);\, x(n)\} \triangleq \frac{\mathrm{PARCOR}\{x(n-m-1);\, x(n)\}}{\sqrt{E\{|e_m^{\mathrm{b}}(n-1)|^2\}E\{|e_m^{\mathrm{f}}(n)|^2\}}} \tag{6.5.21}$$

and, therefore, it takes values in the range $[-1,\ 1]$ (Kendall and Stuart 1979).
Working as in Section 6.2, we can show that

$$E\{e_m^{\mathrm{b}}(n-1)e_m^{\mathrm{f}*}(n)\} = \boldsymbol{b}_m^{\mathrm{H}}\boldsymbol{r}_m + r(m+1) = \beta_m \tag{6.5.22}$$

which in conjunction with

$$E\{|e_m^{\mathrm{b}}(n-1)|^2\} = E\{|e_m^{\mathrm{f}}(n)|^2\} = P_m \tag{6.5.23}$$

and (6.4.16), results in

$$k_m = -\frac{\beta_m}{P_m} = -\mathrm{PCC}\{x(n-m-1);\, x(n)\} \tag{6.5.24}$$

That is, for stationary processes the lattice parameters are the negative of the partial autocorrelation sequence and satisfy the relation

$$|k_m| \le 1 \qquad \text{for all } 0 \le m \le M-1 \tag{6.5.25}$$

derived also for (6.4.25) using an alternate approach.
Minimum phase. The roots of the polynomial $A(z)$ are inside the unit circle if and only if

$$|k_m| < 1 \qquad \text{for all } 0 \le m \le M-1 \tag{6.5.26}$$

which implies that the filters with system functions $A(z)$ and $1/A(z)$ are minimum-phase. The strict inequalities (6.5.26) are satisfied if the stationary process $x(n)$ is nonpredictable, which is the case when the Toeplitz autocorrelation matrix **R** is positive definite.

Lattice-ladder optimization. The output of an FIR lattice filter is a nonlinear function of the lattice parameters. Hence, if we try to design an optimum lattice filter by minimizing the MSE with respect to the lattice parameters, we end up with a nonlinear optimization problem (see Problem 6.11). In contrast, the Levinson algorithm leads to a lattice-ladder realization of the optimum filter through the order-recursive solution of a linear optimization problem. This subject is of interest to signal modeling and adaptive filtering (see Chapters 8 and 9).

6.5.3 Parameter Conversions

We have shown that the *M*th-order forward linear predictor of a stationary process $x(n)$ is uniquely specified by a set of linear equations in terms of the autocorrelation sequence and the prediction error filter is minimum-phase. Furthermore, it can be implemented using either a direct-form structure with coefficients $a_1^{(M)},\ a_2^{(M)},\ \ldots,\ a_M^{(M)}$ or a lattice structure with parameters $k_1,\ k_2,\ \ldots,\ k_M$. Next we show how to convert between the following equivalent representations of a linear predictor:

1. Direct-form filter structure: $\{P_M,\ a_1,\ a_2,\ \ldots,\ a_M\}$.
2. Lattice filter structure: $\{P_M,\ k_0,\ k_1,\ \ldots,\ k_{M-1}\}$.
3. Autocorrelation sequence: $\{r(0),\ r(1),\ \ldots,\ r(M)\}$.

The transformation between the above representations is performed using the algorithms shown in Figure 6.7.

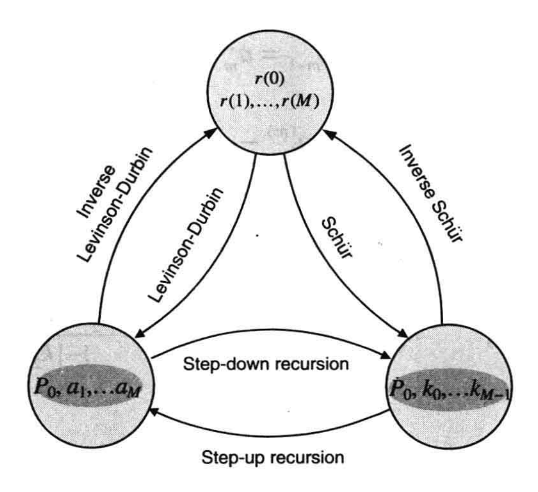


FIGURE 6.7
Equivalent representations for minimum-phase linear prediction error filters.

Lattice-to-direct (step-up) recursion. Given the lattice parameters k_1, k_2, ..., k_M and the MMSE error P_M, we can compute the forward predictor $\boldsymbol{a}_M$ by using the following recursions

$$\boldsymbol{a}_m = \begin{bmatrix} \boldsymbol{a}_{m-1} \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}\boldsymbol{a}^*_{m-1} \\ 1 \end{bmatrix} k_{m-1} \tag{6.5.27}$$

$$P_m = P_{m-1}(1-|k_{m-1}|^2) \tag{6.5.28}$$

for $m=1, 2, \cdots, M$. This conversion is implemented by the function `a=stepup(k) (k,PM)`.

Direct-to-lattice (step-down) recursion. Using the partitioning

$$\begin{aligned} \bar{\mathbf{a}}_m &= [a_1^{(m)}\ a_2^{(m)}\ \cdots\ a_{m-1}^{(m)}]^{\mathrm{T}} \\ k_{m-1} &= a_m^{(m)} \end{aligned} \tag{6.5.29}$$

we can write recursion (6.5.27) as

$$\bar{\boldsymbol{a}}_m = \boldsymbol{a}_{m-1} + \boldsymbol{J}\boldsymbol{a}^*_{m-1} k_{m-1}$$

or by taking the complex conjugate and multiplying both sides by $\mathbf{J}$

$$\boldsymbol{J}\bar{\boldsymbol{a}}^*_m = \boldsymbol{J}\boldsymbol{a}^*_{m-1} + \boldsymbol{a}_{m-1} k^*_{m-1}$$

Eliminating $\boldsymbol{J}\boldsymbol{a}^*_{m-1}$ from the last two equations and solving for $\boldsymbol{a}_{m-1}$, we obtain

$$\boldsymbol{a}_{m-1} = \frac{\bar{\boldsymbol{a}}_m - \boldsymbol{J}\bar{\boldsymbol{a}}^*_m k_{m-1}}{1-|k_{m-1}|^2} \tag{6.5.30}$$

From (6.5.28), we have

$$P_{m-1} = \frac{P_m}{1-|k_{m-1}|^2} \tag{6.5.31}$$

Given $\mathbf{a}_M$ and P_M, we can obtain k_m and P_m for $0 \le m \le M-1$ by computing the last two recursions for $m = M, M-1, \cdots, 2$. We should stress that both recursions break down if $|k_m| = \pm 1$. The step-down algorithm is implemented by the function `[k,P0]=stepdown(a,PM)`.

Example 6.5.1. Given the third-order FLP coefficients $a_1^{(3)}, a_2^{(3)}, a_3^{(3)}$, compute the lattice parameters k_0, k_1, k_2.
Solution. With the help of (6.5.29) the vector relation (6.5.30) can be written in scalar form as

$$k_{m-1} = a_m^{(m)} \tag{6.5.32}$$

and

$$a_i^{(m-1)} = \frac{a_i^{(m)} - a_{m-i}^{(m)*} k_{m-1}}{1 - |k_{m-1}|^2} \tag{6.5.33}$$

which can be used to implement the step-down algorithm for $m=M, M-1, \ldots, 2$ *and* $i = 1, 2, \ldots, m$*.Startingwith* $m = 3$ *and* i = 1, 2, we have

$$k_2 = a_3^{(3)} \qquad a_1^{(2)} = \frac{a_1^{(3)} - a_2^{(3)*}k_2}{1-|k_2|^2} \qquad a_2^{(2)} = \frac{a_2^{(3)} - a_1^{(3)*}k_2}{1-|k_2|^2}$$

Similarly, for m=2 and i=1, we obtain

$$k_1 = a_2^{(2)} \qquad a_1^{(1)} = \frac{a_1^{(2)} - a_1^{(2)*}k_1}{1-|k_1|^2} = k_0$$

which completes the solution.

The step-up and step-down recursions also can be expressed in polynomial form as

$$A_m(z) = A_{m-1}(z) + k_{m-1}^* z^{-m} A_{m-1}^*\left(\frac{1}{z^*}\right) \tag{6.5.34}$$

and

$$A_{m-1}(z) = \frac{A_m(z) - k_{m-1}^* z^{-m} A_m^*(1/z^*)}{1-|k_{m-1}|^2} \tag{6.5.35}$$

respectively

Lattice parameters to autocorrelation. If we know the lattice parameters $k_1, k_2, \ldots, k_M$ and P_M, we can compute the values $r(0), r(1), \ldots, r(M)$ of the autocorrelation sequence using the formula

$$r(m+1) = -k_m^* P_m - \boldsymbol{a}_m^{\mathrm{H}} \boldsymbol{J} \boldsymbol{r}_m \tag{6.5.36}$$

which follows from (6.4.16) and (6.4.17), in conjunction with (6.5.27) and (6.4.21) for $m = 1, 2, \cdots, M$. Equation (6.5.36) is obtained by eliminating β_m from (6.4.9) and (6.4.10). This algorithm is used by the function `r=k2r(k,PM)`.

EXAMPLE 6.5.2. Given P_0, k_0, k_1, and k_2, compute the autocorrelation values $r(0)$, $r(1)$, $r(2)$, and $r(3)$.

Solution. Using $r(0) = P_0$ and

$$r(m+1) = -k_m^* P_m - \boldsymbol{a}_m^{\mathrm{H}} \boldsymbol{J} \boldsymbol{r}_m$$

for $m = 0$, we have

$$r(1) = -k_0^* P_0$$

For $m = 1$

$$r(2) = -k_1^* P_1 - a_1^{(1)*} r(1)$$

where

$$P_1 = P_0(1 - |k_0|^2)$$

Finally, for m=2 we obtain

$$r(3) = -k_2^* P_2 - [a_1^{(2)*} r(2) + k_1^* r(1)]$$

where

$$P_2 = P_1(1 - |k_1|^2)$$

and from the levinson recursion

$$a_1^{(2)} = a_1^{(1)} + a_1^{(1)*} k_1 = k_0 + k_0^* k_1$$

Direct parameters to autocorrelation. Given $\boldsymbol{a}_M$ and P_M, we can compute the autocorrelation sequence $r(0), r(1), \cdots, r(M)$ by using (6.5.29) through (6.5.36). This method is known as the *inverse Levinson algorithm* and is implemented by the function `r=a2r(a,PM)`.

6.6 Summary

The application of optimum FIR filters and linear combiners involves the following two steps.

- *Design*. In this step, we determine the optimum values of the estimator parameters by solving the normal equations formed by using the *known* second-order moments. For stationary processes the design step is done only once. For nonstationary processes, we repeat the design when the statistics change.
- *Implementation*. In this step, we use the optimum parameters and the input data to compute the optimum estimate.

The type and complexity of the algorithms and structures available for the design and implementation of linear MMSE estimators depend on two factors:

- The shift invariance of the input data vector.
- The stationarity of the signals that determine the second-order moments in the normal equations.

As we introduce more structure (shift invariance or stationarity), the algorithms and structures become simpler. From a mathematical point of view, this is reflected in the structure of the correlation matrix, which starting from general Hermitian at one end becomes Toeplitz at the other.

Linear combiners

The input vector is not shift-invariant because the optimum estimate is computed by using samples from $\boldsymbol{M}$ different signals. The correlation matrix $\boldsymbol{R}$ is Hermitian and usually positive definite. The normal equations are solved by using the LDL^H decomposition, and the optimum estimate is computed by using the obtained parameters. However, in many applications where we need the optimum estimate and not the coefficients of the optimum combiner, we can implement the MMSE linear combiner, using the *orthogonal order-recursive structure* shown in Figure 6.1. This structure consists of two parts: (1) a triangular decorrelator (orthogonalizer) that decorrelates the input data vector and produces its innovations vector and (2) a linear combiner that combines the uncorrelated innovations to compute the optimum estimates for *all orders* $1 \le m \le M$.

FIR filters and predictors

In this case the input data vector is shift-invariant, which leads to simplifications, whose extent depends on the stationarity of the involved signals.

Nonstationary case. In general, the correlation matrix is Hermitian and positive definite with no additional structure, and the LDL^H decomposition is the recommended method to solve the normal equations. However, the input shift invariance leads to a remarkable coupling between FLP, BLP, and FIR filtering, resulting in a simplified orthogonal order-recursive structure, which now takes the form of a lattice ladder filter (see Figure 6.3). The backward prediction errors of all orders $1 \le m \le M$ provide the innovations of the input data vector. The parameters of lattice structure (decorrelator) are specified by the components of the LDL^H decomposition of the input correlation matrix. The coefficients of the ladder part (correlator) depend on both the input correlation matrix and the cross-correlation between the desired response and the input data vector.

Stationary case. In this case, the addition of stationarity to the shift invariance makes the correlation matrix Toeplitz. The presence of the Toeplitz structure has the following consequences:

1. The development of efficient order-recursive algorithms, with computational complexity proportional to M^2, for the solution of the normal equations and the triangularization of the correlation matrix.
   a. Levinson algorithm solves $\boldsymbol{Rc} = \boldsymbol{d}$ for arbitrary right-hand side vector $\boldsymbol{d}$ ($2M^2$ operations).
   b. Levinson-Durbin algorithm solves $\boldsymbol{Ra} = -\boldsymbol{r}^*$ when the right-hand side has special structure (M^2 operations).

2. The MMSE FLP, BLP, and FIR filters are time-invariant; that is, their coefficients (direct-form or lattice-ladder structures) are constant and should be computed only once.

   The algorithms for MMSE filtering and prediction of stationary processes are the simplest ones.

However, we can also develop efficient algorithms for nonstationary processes that have special structure. There are two cases of interest:

- The Kalman filtering algorithm that can be used for processes generated by a state-space model with *known* parameters.
- Algorithms for α-stationary processes, that is , processes whose correlation matix is near to Toeplitz, as measured by a special distance known as the *displacement rank* (Morf et al. 1977)

Problems

6.1 By first computing the matrix product

$$\begin{bmatrix} R_m & r_m^b \\ r_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} I_m & 0_m \\ -\dfrac{r_m^{bH}}{\rho_m^b} & \dfrac{1}{\rho_m^b} \end{bmatrix}$$

and then the determinants of both sides, prove Equation (6.1.25).

6.2 Prove the matrix inversion lemma for lower right corner partitioned matrices, which is described by Equations (6.1.26) and (6.1.28).

6.3 This problem generalizes the matrix inversion lemmas to nonsymmetric matrices.

(*a*) Show that if R^{-1} exists, the inverse of an upper left corner partitioned matrix is given by

$$\begin{bmatrix} R & r \\ \tilde{r}^T & \sigma \end{bmatrix}^{-1} = \frac{1}{\alpha}\begin{bmatrix} \alpha R^{-1} + wv^T & w \\ v^T & 1 \end{bmatrix}$$

where

$$Rw \triangleq -r$$
$$R^T v \triangleq -\tilde{r}$$
$$\alpha \triangleq \sigma - \tilde{r}^T R^{-1} r = \sigma + v^T r = \sigma + \tilde{r}^T w$$

(*b*) Show that if R^{-1} exists, the inverse of a lower right corner partitioned matrix is given by

$$\begin{bmatrix} \sigma & \tilde{r}^T \\ r & R \end{bmatrix}^{-1} = \frac{1}{\alpha}\begin{bmatrix} 1 & v^T \\ w & \alpha R^{-1} + wv^T \end{bmatrix}$$

where

$$Rw \triangleq -r$$
$$R^T v \triangleq -\tilde{r}$$
$$\alpha \triangleq \sigma - \tilde{r}^T R^{-1} r = \sigma + v^T r = \sigma + \tilde{r}^T w$$

(*c*) Check the validity of the lemmas in parts (*a*) and (*b*), using MATLAB.

6.4 Develop an order-recursive algorithm to solve the linear system in Example 6.1.2, using the lower right corner partitioning lemma (6.1.26).

6.5 In this problem we consider two different approaches for inversion of symmetric and positive definite matrices by constructing an arbitrary fourth-order positive definite correlation matrix R and comparing their computational complexities.

(*a*) Given that the inverse of a lower (upper) triangular matrix is itself lower (upper) triangular, develop an algorithm for triangular matrix inversion.

(*b*) Compute the inverse of R, using the algorithm in part (*a*) and Equation (6.1.58).

(*c*) Build up the inverse of R, using the recursion (6.1.24).

(*d*) Estimate the number of operations for each method as a function of order M, and check their validity for $M = 4$, using MATLAB.

6.6 Using the appropriate orthogonality principles and definitions, prove Equation (6.3.32).

6.7 Prove Equations (6.3.36) to (6.3.38), using Equation (6.1.45).

6.8 Working as in Example 5.3.1, develop an algorithm for the upper-lower decomposition of a symmetric positive definite matrix. Then use it to factorize the matrix in Example 5.3.1, and verify your results, using the function `[U,D]=udut(R)`.

6.9 In this problem we explore the meaning of the various quantities in the decomposition $R = U\bar{D}U^H$ of the correlation matrix.

(*a*) Show that the rows of $A = U^{-1}$ are the MMSE estimator of x_m from $x_{m+1}, x_{m+2}, \ldots, x_M$.

(*b*) Show that the decomposition $R = U\bar{D}U^H$ can be obtained by the Gram-Schmidt orthogonalization process, starting with the random variable x_M and ending with x_1, that is, proceeding backward.

6.10 In this problem we clarify the various quantities and the form of the partitionings involved in the UDU^H decomposition, using an m=4 correlation matrix.

(*a*) Prove that the components of the forward prediction error vector are uncorrelated.

(*b*) Writing explicitly the matrix R, identify and express the quantities in Equations (6.3.62) through (6.3.67).

(*c*) Using the matrix R in Example 5.3.1, compute the predictors in (6.3.67) by using the corresponding normal equations, verify your results, comparing them with the rows of matrix A computed directly from the LDL^H decomposition of R^{-1} or the UDU^H decomposition of R (see Table 6.1).

6.11 Given an all-zero lattice filter with coefficients k_0 and k_1, determine the MSE $P(k_0,k_1)$ as a function of the required second-order moments, assumed jointly stationary, and plot the error performance surface. Use the statistics in Example 5.2.1.

6.12 Given the autocorrelation $r(0)=1$, $r(1)=r(2)=1/2$, and $r(3)=1/4$, determine all possible representations for the third-order prediction error filter (see Figure 6.7).

6.13 Repeat Problem 6.12 for $k_0=k_1=k_2=1/3$ and $P_3=(2/3)^3$.

6.14 Use Levinson's algorithm to solve the normal equations $\boldsymbol{Rc}=\boldsymbol{d}$ where $\boldsymbol{R}=Toeplitz\{3,\ 2,\ 1\}$ and $\boldsymbol{d}=[6\ 6\ 2]^T$.

6.15 Consider a random sequence with autocorrelation $\{r(l)\}_0^3=\{1,\ 0.8,\ 0.6,\ 0.4\}$. (*a*) Determine the FLP $\mathbf{a}_m$ and the corresponding error P_m^f for $m=1,\ 2,\ 3$. (*b*) Determine and draw the flow diagram of the third-order lattice prediction error filter.

6.16 Using the Levinson-Durbin algorithm, determine the third-order linear predictor $\mathbf{a}_3$ and the MMSE P_3 for a signal with autocorrelation $r(0)=1, r(1)=r(2)=1/2$, and $r(3)=1/4$.

6.17 Given the autocorrelation sequence $r(0)=1$, $r(1)=r(2)=1/2$, and $r(3)=1/4$, compute the lattice and direct-form coefficients of the prediction error filter, using the althorithm of schiir.

6.18 Determine ρ_1 and ρ_2 so that the matrix $\boldsymbol{R}=Toeplitz\{1,\ \rho_1,\ \rho_2\}$ is positive definite.

6.19 Suppose that we want to fit an AR(2) model to a sinusoidal signal with random phase in additive noise. The autocorrelation sequence is given by

$$r(l)=P_0\cos\omega_0 l+\sigma_v^2\delta(l)$$

(*a*) Determine the model parameters $a_1^{(2)}, a_2^{(2)}$, and σ_ω^2 in terms of P_0, ω_0, and σ_v^2. (*b*) Determine the lattice parameters of the model. (*c*) What are the limiting values of the direct and lattice parameters of the model when $\sigma_v^2\to 0$?

6.20 Given the parameters $r(0)=1$, $k_0=k_1=1/2$, and $k_2=1/4$, determine all other equivalent representations of the prediction error filter (see Figure 6.7).

6.21 Let $\{r(l)\}_0^P$ be samples of the autocorrelation sequence of a stationary random signal $x(n)$. (*a*) Is it possible to extend $r(l)$ for $|l|>P$ so that the PSD

$$R(e^{j\omega})=\sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}$$

is valid, that is, $R(e^{j\omega})\geq 0$? (*b*) Using the algorithm of Levinson-Durbin, develop a procedure to check if a given autocorrelation extension is valid. (*c*) Use the algorithm in part (*b*) to find the necessary and sufficient conditions so that $r(0)=1$, $r(1)=\rho_1$, and $r(2)=\rho_2$ are a valid autocorrelation sequence. Is the resulting extension unique?

6.22 Justify the following statements. (*a*) The whitening filter for a stationary process $x(n)$ is time-varying. (*b*) The filter in part (*a*) can be implemented by using a lattice structure and switching its stages on one by one with the arrival of each new sample. (*c*) If *x*(*n*) is AR(*P*), the whitening filter becomes time-invariant *P*+1 sampling intervals after the first sample is applied. *Note:* We assume that the input is applied to the filter at *n*=0. If the input is applied at $n=-\infty$, the whitening filter of a stationary process is always time-invariant.

6.23 Given the parameters $r(0)=1$, $k_0=1/2$, $k_1=1/3$, and $k_2=1/4$, compute the determinant of the matrix $\boldsymbol{R_4}$=*Roeplitz* $\{r(0),\ r(1),\ r(2),\ r(3)\}$.

6.24 (*a*) Determine the lattice second-order prediction error filter (PEF) for a sequence *x*(*n*) with autocorrelation $r(l)=(1/2)^{|l|}$. (*b*) Repeat part (*a*) for the sequence $y(n)=x(n)+v(n)$, where $v(n)\sim WN(0,0.2)$ is uncorrelated to *x*(*n*). (*c*) Explain the change in the lattice parameters using frequency domain reasoning (think of the PEF as a whitening filter).

6.25 Consider a prediction error filter specified by $P_3=(15/16)^2$, $k_0=1/4$, $k_1=1/2$, and $k_2=1/4$. (*a*) Determine the direct-form filter coefficients. (*b*) Determine the autocorrelation values $r(1)$, $r(2)$, and $r(3)$. (*c*) Determine the value $r(4)$ so that the MMSE P_4 for the corresponding fourth-order filter is the minimum possible.

6.26 Consider a prediction error filter $A_M(z)=1+a_1^{(M)}z^{-1}+\cdots+a_M^{(M)}z^{-M}$ with lattice parameters $k_1,\ k_2,\ ...,\ k_M$. (*a*) Show that if we set $\hat{k}_m=(-1)^m k_m$, then $\hat{a}_m^{(M)}=(-1)^m a_m^{(M)}$. (*b*) What are the new filter coefficients if we set $\hat{k}_m=\rho^m k_m$, where ρ is a complex number with $|\rho|=1$? What happens if $|\rho|<1$?

6.27 Show that the MMSE linear predictor of $x(n+D)$ in terms of $x(n),\ x(n-1),\ \cdots,\ x(n-M+1)$ for $D\geq 1$ is given by

$$\boldsymbol{Ra}^{(D)}=-\boldsymbol{r}^{(D)}$$

where $\mathbf{r}^{(D)} = [r(D)\ r\ (D+1)\ \cdots\ r(D+M-1)]^{\mathrm{T}}$. Develop a recursion that computes $\mathbf{a}^{(D+1)}$ from $\mathbf{a}^{(D)}$ by exploring the shift invariance of the vector $\mathbf{r}^{(D)}$. See Manolakis et al. (1983).

6.28 Consider the estimation of a constant α from its noisy observations. The signal and observation models are given by

$$y(n+1) = y(n) \qquad n > 0 \qquad y(0) = \alpha$$
$$x(n) = y(n) + v(n) \qquad v(n) \sim WGN(0, \sigma_v^2)$$

(*a*) Develop scalar Kalman filter equations, assuming the initial condition on the a posteriori error variance $R_{\tilde{y}}(0\,|\,0)$ equal to r_0.
(*b*) Show that the a posteriori error variance $R_{\tilde{y}}(n\,|\,n)$ is given by

$$R_{\tilde{y}}(n\,|\,n) = \frac{r_0}{1 + (r_0 / \sigma_v^2)n} \tag{P.1}$$

(*c*) Show that the optimal filter for the estimation of the constant α is given by

$$\hat{y}(n) = \hat{y}(n-1) + \frac{r_0 / \sigma_v^2}{1 + (r_0 / \sigma_v^2)n}[x(n) - \hat{y}(n-1)]$$

6.29 Consider a random process with PSD given by

$$R_s(e^{j\omega}) = \frac{4}{2.4661 - 1.629\cos\omega + 0.81\cos 2\omega}$$

(*a*) Using MATLAB, plot the PSD $R_s(e^{j\omega})$ and determine the resonant frequency ω_0.
(*b*) Using spectral factorization, develop a signal model for the process of the form

$$\mathbf{y}(n) = \mathbf{A}\mathbf{y}(n-1) + \mathbf{B}\eta(n)$$
$$s(n) = [1 \quad 0]\mathbf{y}(n)$$

where $\mathbf{y}(n)$ is a 2×1 vector, $\eta(n) \sim WGN(0,1)$, and $\mathbf{A}$ and $\mathbf{B}$ are matrices with appropriate dimensions.
(*c*) Let $x(n)$ be the observed values of $s(n)$ given by

$$x(n) = s(n) + v(n) \qquad v(n) \sim WGN(0,1)$$

Assuming reasonable initial conditions, develop Kalman filter equations and implement them, using MATLAB. Study the performance of the filter by simulating a few sample functions of the signal process $s(n)$ and its observation $x(n)$.

6.30 Alternative form of the Kalman filter. A number of different identities and expressions can be obtained for the quantities defining the Kalman filter.

(*a*) By manipulating the last two equations in (6.6.39) show that

$$\begin{aligned} \mathbf{R}_{\tilde{y}}(n\,|\,n) = {} & \mathbf{R}_{\tilde{y}}(n\,|\,n-1) - \mathbf{R}_{\tilde{y}}(n\,|\,n-1)\mathbf{H}^{\mathrm{H}}(n) \\ & \times[\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n\,|\,n-1)\mathbf{H}^{\mathrm{H}}(n) + \mathbf{R}_v(n)]^{-1}\mathbf{H}\mathbf{R}_{\tilde{y}}(n\,|\,n-1) \end{aligned} \tag{P.2}$$

(*b*) If the inverses of $\mathbf{R}_{\tilde{y}}(n\,|\,n)$, $\mathbf{R}_{\tilde{y}}(n\,|\,n-1)$, and $\mathbf{R}_v$ exist, then show that

$$\mathbf{R}_{\tilde{y}}^{-1}(n\,|\,n) = \mathbf{R}_{\tilde{y}}^{-1}(n\,|\,n-1) + \mathbf{H}^{\mathrm{H}}(n)\mathbf{R}_v^{-1}(n)\mathbf{H}(n) \tag{P.3}$$

This shows that the update of the error covariance matrix does not require the Kalman gain matrix (but does require matrix inverses).

(*c*) Finally show that the gain matrix is given by

$$\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n\,|\,n)\mathbf{H}^{\mathrm{H}}(n)\mathbf{R}_v^{-1}(n) \tag{P.4}$$

which is computed by using the a posteriori error covariance matrix.

6.31 In Example 6.6.3 we assumed that only the position measurements were available for estimation. In this problem we will assume that we also have a noisy sensor to measure velocity measurements. Hence the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \end{bmatrix} \tag{P.5}$$

where $v_1(n)$ and $v_2(n)$ are two independent zero-mean white Gaussian noise sources with variances $\sigma_{v_1}^2$ and $\sigma_{v_2}^2$, respectively.

(*a*) Using the state vector model given in Example 6.6.3 and the observation model in (P.5), develop Kalman filter equations to

estimate position and velocity of the object at each n.

(*b*) Using the parameter values

$$T = 0.1 \qquad \sigma_{v_1}^2 = \sigma_{v_2}^2 = \sigma_\eta^2 = 0.25 \qquad y_p(-1) = 0 \qquad y_v(-1) = 1$$

simulate the true and observed positions and velocities of the object. Using your Kalman filter equations, generate plots similar to the ones given in Figures 6.11 and 6.12.

(c) Discuss the effects of velocity measurements on the estimates.

6.32 In this problem, we will assume that the acceleration $y_a(n)$ is an AR(1) process rather than a white noise process. Let $y_a(n)$ be given by

$$y_a(n) = \alpha y_a(n-1) + \eta(n) \qquad \eta(n) \sim WGN(0, \sigma_\eta^2) \qquad y_a(-1) = 0 \tag{P.6}$$

(*a*) Augment the state vector $\mathbf{y}(n)$ in (6.6.48), using variable $y_a(n)$, and develop the state vector as well as the observation model, assuming that only the position is measured.

(*b*) Using the above model and the parameter values

$$T = 0.1 \qquad \alpha = 0.9 \qquad \sigma_v^2 = \sigma_\eta^2 = 0.25$$
$$y_p(-1) = 0 \qquad y_v(-1) = 1 \qquad y_a(-1) = 0$$

simulate the linear motion of the object. Using Kalman filter equations, estimate the position, velocity, and acceleration values of the object at each n. Generate performance plots similar to the ones given in Figures 6.11 and 6.12.

(c) Now assume that noisy measurements of $y_v(n)$ and $y_a(n)$ are also available, that is, the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \\ x_a(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \\ y_a(n) + v_3(n) \end{bmatrix} \tag{P.7}$$

where $v_1(n), v_2(n)$, and $v_3(n)$ are IID zero-mean white Gaussian noise sources with variance σ_v^2. Repeat parts (*a*) and (*b*) above.

CHAPTER 7

Least-Squares Filtering and Prediction

In this chapter, we deal with the design and properties of linear combiners, finite impulse response (FIR) filters, and linear predictors that are optimum in the least-squares error (LSE) sense. The principle of least squares is widely used in practice because second-order moments are rarely known. In the first part of this chapter (Sections 7.1 through 7.4), we concentrate on the design, properties, and applications of least-squares (LS[1]) estimators. Section 7.1 discusses the principle of LS estimation. The unique aspects of the different implementation structures, starting with the general linear combiner followed by the FIR filter and predictor, are treated in Sections 7.2 to 7.4. In the second part (Sections 7.5), we discuss the algorithms for the solution of the LSE normal equations and the computation of LSE estimates including QR decomposition techniques (Householder reflections, Givens rotations, and modified Gram-Schmidt orthogonalization) and the singular value decomposition (SVD).

7.1 The Principle of Least Squares

The *principle of least squares* was introduced by the German mathematician Carl Friedrich Gauss, who used it to determine the orbit of the asteroid Ceres in 1821 by formulating the estimation problem as an optimization problem.

The design of optimum filters in the minimum mean square error (MMSE) sense, discussed in Chapter 5 requires the a priori knowledge of second-order moments. However, such statistical information is simply *not* available in most practical applications, for which we can only obtain measurements of the input and desired response signals. To avoid this problem, we can (1) estimate the required second-order moments from the available data (see Chapter 4, if possible, to obtain an estimate of the optimum MMSE filter, or (2) design an optimum filter by minimizing a criterion of performance that is a function of the available data.

In this chapter, we use the minimization of the sum of the squares of the estimation error as the criterion of performance for the design of optimum filters. This method, known as *least-squares error* (LSE) *estimation*, requires the measurement of *both* the input signal and the desired response signal. A natural question arising at this point is, What is the purpose of estimating the values of a *known*, desired response signal? There are several answers:

1. In system modeling applications, the goal is to obtain a mathematical model describing the input-output behavior of an actual system. A quality estimator provides a good model for the system. The desired result is the estimator or system model, not the actual estimate.
2. In linear predictive coding, the useful result is the prediction error or the respective predictor coefficients.
3. In many applications, the desired response is not available (e.g., digital communications). Therefore, we do not always have a complete set of data from which to design the LSE estimator. However, if the data do not change significantly over a number of sets, then one special complete set, the training set, is used to design the estimator. The resulting estimator is then applied to the processing of the remaining incomplete sets.

The use of measured signal values to determine the coefficients of the estimator leads to some fundamental differences between MMSE and LSE estimation that are discussed where appropriate.

To summarize, depending on the available information, there are two ways to design an optimum estimator: (1) If we know the second-order moments, we use the MMSE criterion and design a filter that is optimum for all possible sets of data with the same statistics. (2) If we only have a block of data, we use the LSE criterion to design an estimator that is optimum for the given block of data. Optimum MMSE estimators are obtained by using ensemble averages, whereas LSE estimators are obtained by using finite-length time averages. For example, an MMSE estimator, designed using ensemble averages, is optimum for all realizations. In contrast, an LSE estimator, designed

[1]A note about abbreviations used throughout the chapter: The two acronyms *LSE* and *LS* will be used almost interchangably. Although LSE is probably the more accurate term, LS has become a standard reference to LSE estimators.

using a block of data from a particular realization, depends on the numerical values of samples used in the design. If the processes are ergodic, the LSE estimator approaches the MMSE estimator as the block length of the data increases toward infinity.

7.2 Linear Least-Squares Error Estimation

We start with the derivation of general linear LS filters that are implemented using the linear combiner structure described in Section 5.2. A set of measurements of the desired response $y(n)$ and the input signals $x_k(n)$ for $1 \le k \le M$ has been taken for $0 \le n \le N-1$. As in optimum MMSE estimation, the problem is to estimate the desired response $y(n)$ using the linear combination

$$\hat{y}(n) = \sum_{k=1}^{M} c_k^*(n)\, x_k(n) = \mathbf{c}^{\mathrm{H}}(n)\,\mathbf{x}(n) \tag{7.2.1}$$

We define the estimation error as

$$e(n) = y(n) - \hat{y}(n) = y(n) - \mathbf{c}^{\mathrm{H}}(n)\,\mathbf{x}(n) \tag{7.2.2}$$

and the coefficients of the combiner are determined by minimizing the sum of the squared errors

$$\boldsymbol{E} \triangleq \sum_{n=0}^{N-1} |e(n)|^2 \tag{7.2.3}$$

that is, the *energy* of the error signal. *For this minimization to be possible, the coefficient vector* $\mathbf{c}(n)$ *should be held constant over the measurement time interval* $0 \le n \le N-1$. The constant vector $\mathbf{c}_{ls}$ resulting from this optimization depends on the measurement set and is known as the *linear LSE estimator.* In the statistical literature, LSE estimation is known as linear regression, where (7.2.2) is called a *regression function,* $e(n)$ are known as *residuals* (leftovers), and $\mathbf{c}(n)$ is the *regression vector* (Montgomery and Peck 1982).

The system of equations in (7.2.2), or equivalently $e^*(n) = y^*(n) - \mathbf{x}^{\mathrm{H}}(n)\mathbf{c}$, can be written in matrix form as

$$\begin{bmatrix} e^*(0) \\ e^*(1) \\ \vdots \\ e^*(N-1) \end{bmatrix} = \begin{bmatrix} y^*(0) \\ y^*(1) \\ \vdots \\ y^*(N-1) \end{bmatrix} - \begin{bmatrix} x_1^*(0) & x_2^*(0) & \cdots & x_M^*(0) \\ x_1^*(1) & x_2^*(1) & \cdots & x_M^*(1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1^*(N-1) & x_2^*(N-1) & \cdots & x_M^*(N-1) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} \tag{7.2.4}$$

or more compactly as

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{c} \tag{7.2.5}$$

where

$$\begin{aligned} \boldsymbol{e} &\triangleq [e(0)\ e(1)\ \cdots\ e(N-1)]^{\mathrm{H}} && \text{error data vector } (N \times 1) \\ \boldsymbol{y} &\triangleq [y(0)\ y(1)\ \cdots\ y(N-1)]^{\mathrm{H}} && \text{desired response vector } (N \times 1) \\ \boldsymbol{X} &\triangleq [\boldsymbol{x}(0)\ \boldsymbol{x}(1)\ \cdots\ \boldsymbol{x}(N-1)]^{\mathrm{H}} && \text{input data matrix } (N \times M) \\ \boldsymbol{c} &\triangleq [c_1 c_2 \cdots c_M]^{\mathrm{T}} && \text{combiner parameter vector } (M \times 1) \end{aligned} \tag{7.2.6}$$

are defined by comparing (7.2.4) to (7.2.5). The input data matrix $\mathbf{X}$ can be partitioned either columnwise or rowwise as follows:

$$\boldsymbol{X} \triangleq [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_M] = \begin{bmatrix} \boldsymbol{x}^{\mathrm{H}}(0) \\ \boldsymbol{x}^{\mathrm{H}}(1) \\ \vdots \\ \boldsymbol{x}^{\mathrm{H}}(N-1) \end{bmatrix} \tag{7.2.7}$$

where the columns $\tilde{\boldsymbol{x}}_k$ of $\boldsymbol{X}$

$$\tilde{\boldsymbol{x}}_k \triangleq [x_k(0)\ x_k(1)\ \cdots\ x_k(N-1)]^{\mathrm{H}}$$

will be called *data records* and the rows

$$\mathbf{x}(n) \triangleq [x_1(n)\ x_2(n)\ \cdots\ x_M(n)]^T$$

will be called *snapshots*. Both of these partitionings of the data matrix, which are illustrated in Figure 7.1, are useful in the derivation, interpretation, and computation of LSE estimators.

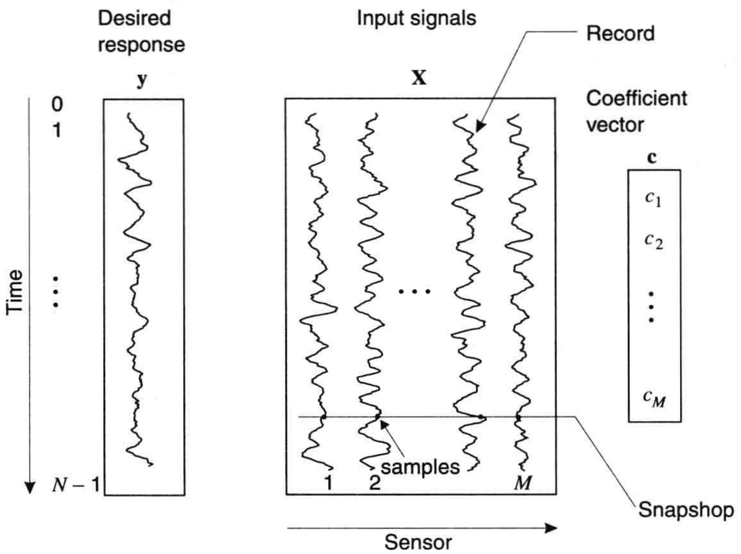


FIGURE 7.1
The columns of the data matrix are the records of data collected at each input (sensor), whereas each row contains the samples from all inputs at the same instant.

The LSE estimator operates in a block processing mode; that is, it processes a frame of N snapshots using the steps shown in Figure 7.2. The input signals are blocked into frames of N snapshots with successive frames overlapping by N_0 samples. The values of N and N_0 depend on the application. The required estimate or residual signals are unblocked at the final stage of the processor.

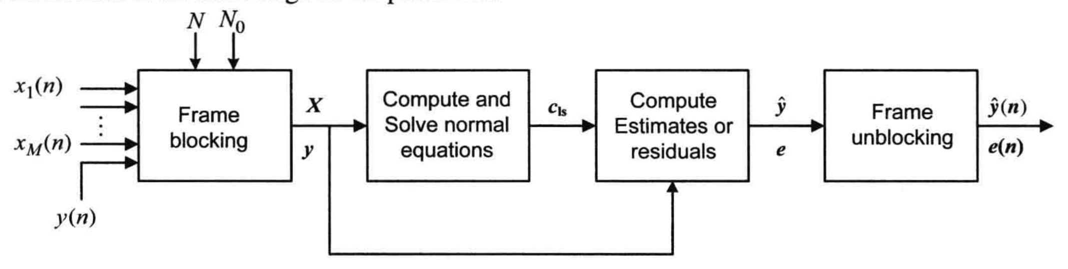


FIGURE 7.2
Block processing implementation of a general linear LSE estimator.

If we set $e = 0$, we have a set of N equations with M unknowns. If $N = M$, then (7.2.4) usually has a unique solution. For $N > M$, we have an overdetermined system of linear equations that typically has no solution. Conversely, if $N < M$, we have an underdetermined system that has an infinite number of solutions. However, even if $M > N$ or $N > M$, the system (7.2.4) has a natural, unique, least-squares solution. We next focus our attention on overdetermined systems since they play a very important role in practical applications.

7.2.1 Derivation of the Normal Equations

We provide an algebraic and a geometric solution to the LSE estimation problem; a calculus-based derivation is given in Problem 7.1.

Algebraic derivation. The energy of the error can be written as

$$\begin{aligned} E = e^{\mathrm{H}} e &= (y^{\mathrm{H}} - c^{\mathrm{H}} X^{\mathrm{H}})(y - Xc) \\ &= y^{\mathrm{H}} y - c^{\mathrm{H}} X^{\mathrm{H}} y - y^{\mathrm{H}} Xc + c^{\mathrm{H}} X^{\mathrm{H}} Xc \\ &= E_y - c^{\mathrm{H}} \hat{d} - \hat{d}^{\mathrm{H}} c + c^{\mathrm{H}} \hat{R} c \end{aligned} \tag{7.2.8}$$

where

$$E_y \triangleq y^{\mathrm{H}} y = \sum_{n=0}^{N-1} |y(n)|^2 \tag{7.2.9}$$

$$\hat{R} \triangleq X^{\mathrm{H}} X = \sum_{n=0}^{N-1} x(n) x^{\mathrm{H}}(n) \tag{7.2.10}$$

$$\hat{d} \triangleq X^{\mathrm{H}} y = \sum_{n=0}^{N-1} x(n) y^*(n) \tag{7.2.11}$$

Note that these quantities can be viewed as time-average estimates of the desired response power, correlation matrix of the input data vector, and the cross-correlation vector between the desired response and the data vector, when these quantities are divided by the number of data samples N.

We emphasize that all formulas derived for the MMSE criterion hold for the LSE criterion if we replace the expectation operator $E\{(\cdot)\}$ with the time-average operator $\sum_{n=0}^{N-1}(\cdot)$. This results from the fact that both criteria are quadratic cost functions. Therefore, working as in Section 5.2.2, we conclude that if the time-average correlation matrix $\hat{R}$ is positive definite, the LSE estimator $\mathbf{c}_{ls}$ is provided by the solution of the normal equations

$$\hat{R} c_{ls} = \hat{d} \tag{7.2.12}$$

and the minimum sum of squared errors is given by

$$E_{ls} = E_y - \hat{d}^{\mathrm{H}} \hat{R}^{-1} \hat{d} = E_y - \hat{d}^{\mathrm{H}} c_{ls} \tag{7.2.13}$$

Since R is Hermitian, we only need to compute the elements

$$\hat{r}_{ij} = \tilde{x}_i^{\mathrm{H}} \tilde{x}_j \tag{7.2.14}$$

in the upper triangular part, which requires $M(M+1)/2$ dot products. The right-hand side requires M dot products

$$\hat{d}_i = \tilde{x}_i^{\mathrm{H}} y \tag{7.2.15}$$

Note that each dot product involves N arithmetic operations, each consisting of one multiplication and one addition. Thus, to form the normal equations requires a total of

$$\frac{1}{2} M(M+1)N + MN = \frac{1}{2} M^2 N + \frac{3}{2} MN \tag{7.2.16}$$

arithmetic operations. When R is nonsingular, which is the case when R is positive definite, we can solve the normal equations using either the LDL^H or the Cholesky decomposition (see Section 5.3). However, it should be stressed at this point that most of the computational work lies in forming the normal equations rather than their solution. The formulation of the overdetermined LS equations and the normal equations is illustrated graphically in Figure 7.3. The solution of LS problems has been extensively studied in various application areas and in numerical analysis. The basic methods for the solution of the LS problem, which are discussed in this book, are shown in Figure 7.4. We just stress here that for overdetermined LS problems, well-behaved data, and sufficient numerical precision, all these methods provide comparable results.

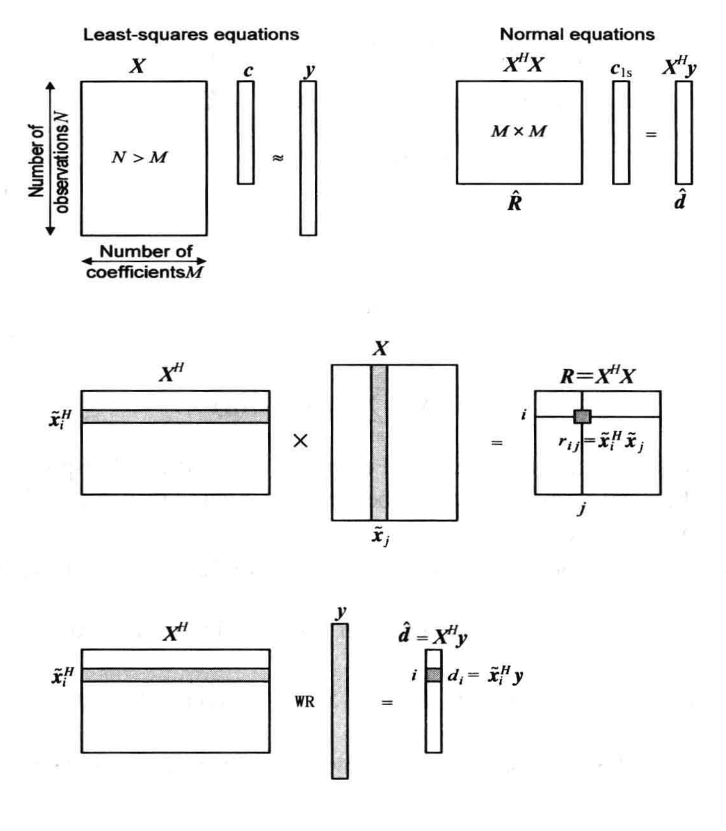


FIGURE 7.3
The LS problem and computation of the normal equations.

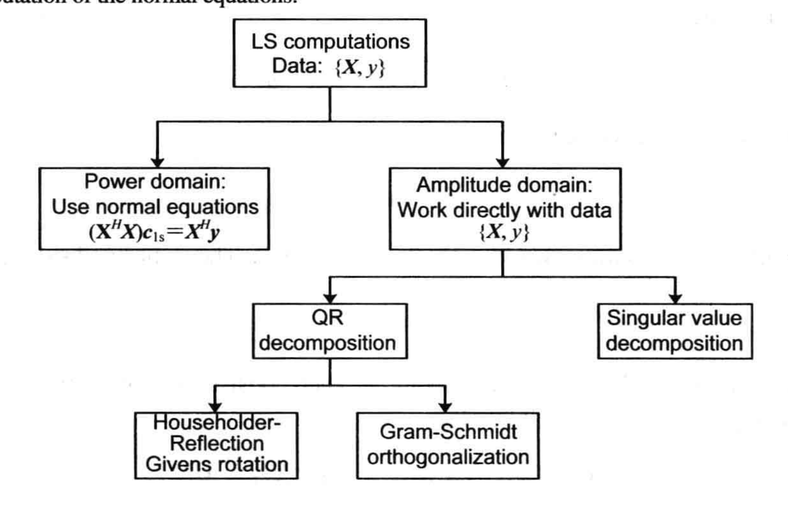


FIGURE 7.4
Classification of different computational algorithms for the solution of the LS problem.

Geometric derivation. We may think of the desired response record $\mathbf{y}$ and the data records $\tilde{\mathbf{x}}_k$, $1 \leq k \leq M$, as vectors in an N-dimensional vector space, with the dot product and length defined by

$$<\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j> \triangleq \tilde{\boldsymbol{x}}_i^H \tilde{\boldsymbol{x}}_j = \sum_{n=0}^{N-1} x_i(n) x_j^*(n) \tag{7.2.17}$$

and

$$\|\tilde{\boldsymbol{x}}\|^2 \triangleq <\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}> = \sum_{n=0}^{N-1} |x(n)|^2 = E_x \tag{7.2.18}$$

respectively. The estimate of the desired response record can be expressed as

$$\hat{y} = Xc = \sum_{k=1}^{M} c_k \tilde{x}_k \tag{7.2.19}$$

that is, as a linear combination of the data records.

The M vectors $\tilde{x}_k$ form an M-dimensional subspace, called the *estimation space*, which is the column space of data matrix X. Clearly, any estimate $\hat{y}$ must lie in the estimation space. The desired response record y, in general, lies outside the estimation space. The estimation space for $M = 2$ and $N = 3$ is illustrated in Figure 7.5. The error vector e points from the tip of $\hat{y}$ to the tip of y. The squared length of e is minimum when e is perpendicular to the estimation space, that is, $e \perp \tilde{x}_k$ for $1 \le k \le M$.

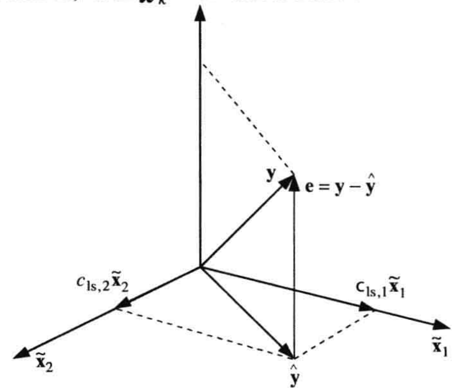


FIGURE 7.5
Vector space interpretation of LSE estimation for $N = 3$ (dimension of data space) and $M = 2$ (dimension of estimation subspace).

Therefore, we have the orthogonality principle

$$< \tilde{x}_k, e > \; = \; \tilde{x}_k^{H} e = 0 \qquad 1 \le k \le M \tag{7.2.20}$$

or more compactly

$$X^{H} e = X^{H}(y - Xc_{ls}) = 0$$

or

$$(X^{H} X) c_{ls} = X^{H} y \tag{7.2.21}$$

which we recognize as the LSE normal equations from (7.2.12).

The LS solution splits the desired response y into two orthogonal components, namely, $\hat{y}_{ls}$ and e_{ls}. Therefore,

$$\|y\|^2 = \|\hat{y}_{ls}\|^2 + \|e_{ls}\|^2 \tag{7.2.22}$$

and, using (7.2.18) and (7.2.19), we have

$$E_{ls} = E_y - \mathbf{c}_{ls}^{H} \mathbf{X}^{H} \mathbf{X} \mathbf{c}_{ls} = E_y - \mathbf{c}_{ls}^{H} \mathbf{X}^{H} \mathbf{y} \tag{7.2.23}$$

which is identical to (7.2.13). The normalized total squared error is

$$\varepsilon \triangleq \frac{E_{ls}}{E_y} = 1 - \frac{E_{\hat{y}}}{E_y} \tag{7.2.24}$$

which is in the range $0 \le \varepsilon \le 1$, with limits of 0 and 1, which correspond to the worst and best cases, respectively.

Uniqueness. The solution of the LSE normal equations exists and is unique if the time-average correlation matrix **R** is invertible. We shall prove the following:

THEOREM 7.1. The time-average correlation matrix $\hat{R} = X^{H} X$ is invertible if and only if the columns $\tilde{x}_k$ of X are linearly independent, or equivalently if and only if R is positive definite.

Proof. If the columns of X are linearly independent, then for every $z \ne 0$ we have $Xz \ne 0$. This implies that for every $z \ne 0$

$$z^{H}(X^{H} X) z = (Xz)^{H} Xz = \|Xz\|^2 > 0 \tag{7.2.25}$$

that is, $\boldsymbol{R}$ is positive definite and hence nonsingular.

If the columns of $\boldsymbol{X}$ are linearly dependent, then there is a vector $z_0 \neq 0$ such that $\boldsymbol{X}z_0 = 0$. Therefore, $\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}z_0 = 0$, which implies that $\hat{\boldsymbol{R}} = \boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}$ is singular.

For a matrix to have linearly independent columns, the number of rows should be equal to or larger than the number of columns; that is, we must have more equations than unknowns. To summarize, *the overdetermined* $(N > M)$ *LS problem has a unique solution provided by the normal equations in (7.2.12) if the time-average correlation matrix* $\boldsymbol{R}$ *is positive definite, or equivalently if the data matrix* $\boldsymbol{X}$ *has linearly independent columns.*

In this case, the LS solution can be expressed as

$$\boldsymbol{c}_{\mathrm{ls}} = \boldsymbol{X}^{+}\boldsymbol{y} \tag{7.2.26}$$

where

$$\boldsymbol{X}^{+} \triangleq (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}} \tag{7.2.27}$$

is an $M \times N$ matrix known as the *pseudo-inverse* or the *Moore-Penrose generalized inverse* of matrix $\mathbf{X}$ (Golub and Van Loan 1996; Strang 1980).

The LS estimate $\hat{\boldsymbol{y}}_{\mathrm{ls}}$ of $\mathbf{y}$ can be expressed as

$$\hat{\boldsymbol{y}}_{\mathrm{ls}} = \boldsymbol{P}\boldsymbol{y} \tag{7.2.28}$$

where

$$\mathrm{P} \triangleq \mathbf{x}(\mathbf{x}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}} \tag{7.2.29}$$

is know as the *projection matrix* because it projects the data vector $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$ to provide the LS estimate $\hat{\boldsymbol{y}}_{\mathrm{ls}}$ of $\boldsymbol{y}$. Similarly, the LS error vector $\boldsymbol{e}_{ls}$ can be expressed as

$$\boldsymbol{e}_{\mathrm{ls}} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y} \tag{7.2.30}$$

where $\boldsymbol{I}$ is the $N \times N$ identity matrix. The projection matrix $\boldsymbol{P}$ is Hermitian and *idempotent*, that is,

$$\boldsymbol{P} = \boldsymbol{P}^{\mathrm{H}} \tag{7.2.31}$$

and

$$\boldsymbol{P}^2 = \boldsymbol{P}^{\mathrm{H}}\boldsymbol{P} = \boldsymbol{P} \tag{7.2.32}$$

respectively.

When the columns of $\boldsymbol{X}$ are linearly dependent, the LS problem has many solutions. Since all these solutions satisfy the normal equations and the orthogonal projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$ is unique, all these solutions produce an error vector $\mathbf{e}$ of equal length, that is, the same LSE. This subject is discussed in Section 7.6.2 (minimum-norm solution).

EXAMPLE 7.2.1 Suppose that we wish to estimate the sequence $\mathbf{y} = [1\ 2\ 3\ 2]^{\mathrm{T}}$ from the observation vectors $\tilde{x}_1 = [1\ 2\ 1\ 1]^T$ and $\tilde{x}_2 = [2\ 1\ 2\ 3]^T$. Determine the optimum filter, the error vector $\boldsymbol{e}_{\mathrm{ls}}$, and the LSE E_{ls}.

Solution. We first compute the quantities

$$\hat{\boldsymbol{R}} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 7 & 9 \\ 9 & 18 \end{bmatrix} \qquad \hat{\boldsymbol{d}} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 16 \end{bmatrix}$$

and we then solve the normal equations $\hat{\boldsymbol{R}}\boldsymbol{c}_{ls} = \hat{\boldsymbol{d}}$ to obtain the LS estimator

$$\boldsymbol{c}_{\mathrm{ls}} = \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}} = \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{7}{45} \end{bmatrix} \begin{bmatrix} 10 \\ 16 \end{bmatrix} = \begin{bmatrix} \frac{4}{5} \\ \frac{22}{45} \end{bmatrix}$$

and the LSE

$$E_{\text{ls}} = E_y - \hat{\boldsymbol{d}}^{\text{T}} \boldsymbol{c}_{\text{ls}} = 18 - \begin{bmatrix} 10 \\ 16 \end{bmatrix}^{\text{T}} \begin{bmatrix} \frac{4}{5} \\ \frac{22}{45} \end{bmatrix} = \frac{98}{45}$$

The projection matrix is

$$\boldsymbol{P} = \boldsymbol{X}(\boldsymbol{X}^{\text{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{T}} = \begin{bmatrix} \frac{2}{9} & \frac{1}{9} & \frac{2}{9} & \frac{1}{3} \\ \frac{1}{9} & \frac{43}{45} & \frac{1}{9} & -\frac{2}{15} \\ \frac{2}{9} & \frac{1}{9} & \frac{2}{9} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{15} & \frac{1}{3} & \frac{3}{5} \end{bmatrix}$$

which can be used to determine the error vector

$$\boldsymbol{e}_{\text{ls}} = \boldsymbol{y} - \boldsymbol{P}\boldsymbol{y} = [-\frac{7}{9} \quad -\frac{4}{45} \quad \frac{11}{9} \quad -\frac{4}{15}]^{\text{T}}$$

whose squared norm is equal to $\|\boldsymbol{e}_{\text{ls}}\|^2 = \frac{98}{45} = E_{\text{ls}}$, as expected. We can also easily verify the orthogonality principle $\mathbf{e}_{\text{ls}}^T \tilde{\mathbf{x}}_1 = \mathbf{e}_{\text{ls}}^T \tilde{\mathbf{x}}_2 = 0$.

Weighted least-squares estimation. The previous results were derived by using an LS criterion that treats every error $e(n)$ equally. However, based on a priori information, we may wish to place greater importance on different errors, using the weighted LS criterion

$$E_\omega = \sum_{n=0}^{N-1} \omega(n) \, | \, e(n) \, |^2 = \boldsymbol{e}^{\text{H}} \boldsymbol{W} \boldsymbol{e} \tag{7.2.33}$$

where

$$\boldsymbol{W} \triangleq \text{diag}\{\omega(0), \ \omega(1), \ \cdots, \ \omega(N-1)\} \tag{7.2.34}$$

is a diagonal weighting matrix with positive elements. Usually, we choose small weights where the errors are expected to be large, and vice versa. Minimization of E_ω with respect to $\mathbf{c}$ yields the *weighted LS (WLS) estimator*

$$\boldsymbol{c}_{\text{wls}} = (\boldsymbol{X}^{\text{H}} \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^{\text{H}} \boldsymbol{W} \boldsymbol{y} \tag{7.2.35}$$

assuming that the inverse of the matrix $\boldsymbol{X}^{\text{H}}\boldsymbol{W}\boldsymbol{X}$ exists. We can easily see that when $\boldsymbol{W} = \boldsymbol{I}$, then $\boldsymbol{c}_{\text{wls}} = \boldsymbol{c}_{\text{ls}}$. The criterion in (7.2.33) can be generalized by choosing $\boldsymbol{W}$ to be any Hermitian, positive definite matrix (see Problem 7.2).

7.2.2 Statistical Properties of Least-Squares Estimators

A useful approach for evaluating the quality of an LS estimator is to study its statistical properties. Toward this end, we assume that the obtained measurements $\mathbf{y}$ actually have been generated by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}_o + \boldsymbol{e}_o \tag{7.2.36}$$

where $\boldsymbol{e}_o$ is the random measurement error vector. We may think of $\boldsymbol{c}_o$ as the "true" parameter vector. Using (7.2.36), we see that (7.2.21) gives

$$\boldsymbol{c}_{\text{ls}} = \boldsymbol{c}_o + (\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{H}}\boldsymbol{e}_o \tag{7.2.37}$$

We make the following assumptions about the random measurement error vector $\boldsymbol{e}_o$:

1. The error vector $\mathbf{e}_o$ has zero mean

$$E\{e_o\} = 0 \tag{7.2.38}$$

2. The error vector $\mathbf{e}_o$ has uncorrelated components with constant variance $\sigma_{e_o}^2$; that is, the correlation matrix is given by

$$R_{e_o} = E\{e_o e_o^{\mathrm{H}}\} = \sigma_{e_o}^2 I \tag{7.2.39}$$

3. There is no information about $\mathbf{e}_o$ contained in data matrix **X**; that is,

$$E\{e_o \mid X\} = E\{e_o\} = \mathbf{0} \tag{7.2.40}$$

4. If **X** is a deterministic $N \times M$ matrix, then it has rank *M*. This means that **X** is a full-column rank and that $X^{\mathrm{H}}X$ is invertible. If *X* is a stochastic $N \times M$ matrix, then $E\{(X^{\mathrm{H}}X)^{-1}\}$ exists.

In the following analysis, we consider two possibilities: *X* is deterministic and stochastic. Under these conditions, the LS estimator c_{ls} has several desirable properties.

Deterministic data matrix

In this case, we assume that the LS estimators are obtained from the deterministic data values; that is, the matrix *X* is treated as a matrix of constants. Then the properties of the LS estimators can be derived from the statistical properties of the random measurement error vector e_o.

PROPERTY 7.2.1 The LS estimator c_{ls} is an unbiased estimator of c_o, that is,

$$E\{c_{ls}\} = c_o$$

Proof. Taking the expectation of both sides of (7.2.37), we have

$$E\{c_{ls}\} = E\{c_o\} + (X^{\mathrm{H}}X)^{-1}X^{\mathrm{H}}E\{e_o\} = c_o \tag{7.2.41}$$

because *X* is deterministic and $E\{e_o\} = 0$.

PROPERTY 7.2.2 The covariance matrix of c_{ls} corresponding to the error $c_{ls} - c_o$ is

$$\Gamma_{\mathrm{ls}} \triangleq E\{(c_{ls} - c_o)(c_{ls} - c_o)^{\mathrm{H}}\} = \sigma_{e_o}^2 (X^{\mathrm{H}}X)^{-1} = \sigma_{e_o}^2 \hat{R}^{-1} \tag{7.2.42}$$

Proof. Using (7.2.37), (7.2.39), and the definition (7.2.42), we easily obtain

$$\Gamma_{\mathrm{ls}} = (X^{\mathrm{H}}X)^{-1}X^{\mathrm{H}}E\{e_o e_o^{\mathrm{H}}\}X(X^{\mathrm{H}}X)^{-1} = \sigma_{e_o}^2 (X^{\mathrm{H}}X)^{-1}$$

Note that the diagonal elements of matrix $\sigma_e^2 \hat{R}^{-1}$ are also equal to the variance of the LS combiner vector c_{ls}.

PROPERTY 7.2.3 An unbiased estimate of the error variance $\sigma_{e_o}^2$ is given by

$$\hat{\sigma}_{e_o}^2 = (E_{\mathrm{ls}} / N - M) \tag{7.2.43}$$

where *N* is the number of observations, *M* is the number of parameters, and E_{ls} is the LS error.

Proof. Using (7.2.30) and (7.2.36), we obtain

$$e_{\mathrm{ls}} = (I - P)y = (I - P)e_o$$

which results in

$$E_{\mathrm{ls}} = e_{\mathrm{ls}}^{\mathrm{H}} e_{\mathrm{ls}} = e_o^{\mathrm{H}}(I - P)^{\mathrm{H}}(I - P)e_o = e_o^{\mathrm{H}}(I - P)e_o$$

because of (7.2.32). Since E_{ls} depends on $\mathbf{e}_o$, it is a random variable whose expected value is

$$E\{E_{\mathrm{ls}}\} = E\{e_o^{\mathrm{H}}(I - P)e_o\} = E\{\mathrm{tr}[(I - P)e_o e_o^{\mathrm{H}}]\}$$
$$= \mathrm{tr}[(I - P)E\{e_o e_o^{\mathrm{H}}\}] = \sigma_e^2 \mathrm{tr}(I - P)$$

since $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$, where tr is the trace function. However,

$$\begin{aligned}\mathrm{tr}(\boldsymbol{I}-\boldsymbol{P}) &= \mathrm{tr}[\boldsymbol{I}-\boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}] \\ &= \mathrm{tr}[\boldsymbol{I}_{N\times N}-(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}] \\ &= \mathrm{tr}(\boldsymbol{I}_{N\times N})-\mathrm{tr}[(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}] \\ &= \mathrm{tr}(\boldsymbol{I}_{N\times N})-\mathrm{tr}(\boldsymbol{I}_{M\times M}) = N-M\end{aligned}$$

therefore

$$\sigma_{e_o}^2 = \frac{E\{E_{\mathrm{ls}}\}}{N-M} \tag{7.2.44}$$

which proves that $\hat{\sigma}_e^2$ is an unbiased estimate of $\sigma_{e_o}^2$.

Similar to (7.2.41), the mean value of $\mathbf{c}_{\mathrm{wls}}$ is

$$E\{\boldsymbol{c}_{wls}\} = E\{\boldsymbol{c}_o\}+(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{W}E\{\boldsymbol{e}_o\} = E\{\boldsymbol{c}_o\} \tag{7.2.45}$$

that is, the WLS estimator is an unbiased estimate of $\boldsymbol{c}_o$. The covariance matrix of $\boldsymbol{c}_{\mathrm{wls}}$ is

$$\Gamma_{\mathrm{wls}} = (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{R}_{e_o}\boldsymbol{W}\boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{W}\boldsymbol{X})^{-1} \tag{7.2.46}$$

where $\boldsymbol{R}_{e_o}$ is the correlation matrix of $\boldsymbol{e}_o$. It is easy to see that when $\boldsymbol{R}_{e_o} = \sigma_{e_o}^2\boldsymbol{I}$ and $\boldsymbol{W} = \boldsymbol{I}$, we obtain (7.2.42).

PROPERTY 7.2.4 The trace of Γ_{wls} attains its minimum when $\boldsymbol{W} = \boldsymbol{R}_{e_o}^{-1}$. The resulting estimator

$$\boldsymbol{c}_{\mathrm{mv}} = (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{R}_{e_o}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{R}_{e_o}^{-1}\boldsymbol{y} \tag{7.2.47}$$

is known as the *minimum variance* or *Markov estimator* and is the *best linear unbiased estimator* (*BLUE*).

Proof. The proof is somewhat involved. Interested readers can see Goodwin and Payne (1977) and Scharf (1991).

PROPERTY 7.2.5 If $\boldsymbol{R}_{e_o} = \sigma_{e_o}^2\boldsymbol{I}$, the LS estimator $\boldsymbol{c}_{\mathrm{ls}}$ is also the best linear unbiased estimator.

Proof. It follows from (7.2.47) with the substitution $\boldsymbol{R}_{e_o} = \sigma_{e_o}^2\boldsymbol{I}$.

PROPERTY 7.2.6 When the random observation vector $\boldsymbol{e}_o$ has a normal distribution with mean zero and correlation matrix $\boldsymbol{R}_{e_o} = \sigma_{e_o}^2\boldsymbol{I}$, that is, when its components are uncorrelated, the LS estimator $\boldsymbol{c}_{\mathrm{ls}}$ is also the maximum likelihood estimator.

Proof. Since the components of vector $\boldsymbol{e}_o$ are uncorrelated and normally distributed with zero mean and variance σ_e^2, the likelihood function for real-valued $\boldsymbol{e}_o$ is given by

$$L(\boldsymbol{c}) = \prod_{n=0}^{N-1}\frac{1}{\sqrt{2\pi}\sigma_{e_o}}\exp\left[-\frac{|e_o(n)|^2}{2\sigma_{e_o}^2}\right] \tag{7.2.48}$$

and its logarithm by

$$\ln L(\boldsymbol{c}) = -\frac{1}{2\sigma_{e_o}^2}\boldsymbol{e}_o^{\mathrm{H}}\boldsymbol{e}_o - \frac{N}{2}\ln(2\pi\sigma_{e_o}^2) = -\frac{1}{2\sigma_{e_o}^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{c})^{\mathrm{H}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{c})+\mathrm{const} \tag{7.2.49}$$

For complex-valued $\boldsymbol{e}_o$, the terms $\sqrt{2\pi}\sigma_{e_o}$ and $2\sigma_{e_o}^2$ in (7.2.48) are replaced by $\pi\sigma_{e_o}^2$ and $\sigma_{e_o}^2$, respectively. Since the logarithm is a monotonic function, maximization of $L(\boldsymbol{c})$ is equivalent to minimization of $\ln L(\boldsymbol{c})$. It is easy to see, by comparison with (7.2.8), that the LS solution maximizes this likelihood function.

Stochastic data matrix

We now extend the statistical properties of c_{ls} from the preceding section to the situation in which the data values in X are obtained from a random source with a known probability distribution. This situation is best handled by first obtaining the desired results conditioned on X, which is equivalent to the deterministic case. We then determine the unconditional results by (statistical) averaging over the conditional distributions using the following properties of the conditional averages.

The *conditional mean* and the *conditional covariance* of a random vector $\mathbf{x}(\zeta)$, given another random vector $\mathbf{y}(\zeta)$, are defined by

$$\boldsymbol{\mu}_{x|y} \triangleq E\{x(\boldsymbol{\zeta}) \mid y(\boldsymbol{\zeta})\}$$

and

$$\boldsymbol{\Gamma}_{x|y} \triangleq E\{[x(\zeta)-\boldsymbol{\mu}_{x|y}][x(\zeta)-\boldsymbol{\mu}_{x|y}]^{\mathrm{H}} \mid y(\zeta)\}$$

respectively. Since both quantities are random objects, it can be shown that

$$\mu_x = E\{x(\zeta)\} = E_y\{E\{x(\zeta) \mid y(\zeta)\}\}$$

which is known as the *law of iterated expectations* and that

$$\boldsymbol{\Gamma}_x = \boldsymbol{\Gamma}^{(y)}_{\mu_{x|y}} + \boldsymbol{\mu}^{(y)}_{\Gamma_{x|y}}$$

which is called the *decomposition of the covariance rule*. This rule states that the covriance of a random vector $x(\zeta)$ decomposes into the covariance of the conditional mean plus the mean of the conditional covariance. The covariance of the conditional mean, $\mu_{x|y}$, is given by

$$\boldsymbol{\Gamma}^{(y)}_{\mu_{x|y}} \triangleq E_y\{[\boldsymbol{\mu}_{x|y}-\boldsymbol{\mu}_x][\boldsymbol{\mu}_{x|y}-\boldsymbol{\mu}_x]^{\mathrm{H}}\}$$

where the notation $\Gamma^{(y)}_{[\cdot]}$ indicates the covariance over the distribution of $y(\zeta)$. More details can be found in Greene (1993).

PROPERTY 7.2.7. The LS estimator $\mathbf{c}_{ls}$ is an unbiased estimator of $\mathbf{c}_o$.

Proof. Taking the conditional expectation with respect to X of both sides of (7.2.37), we obtain

$$E\{c_{\mathrm{ls}} \mid X\} = E\{c_o \mid X\} + (X^{\mathrm{H}}X)^{-1}X^{\mathrm{H}}E\{e_o \mid X\} \tag{7.2.50}$$

Now using the law of iterated expectations, we get

$$E\{c_{ls}\} = E_X\{E\{c_{ls} \mid X\}\} = c_o + E\{(X^{\mathrm{H}}X)^{-1}X^{\mathrm{H}}E\{e_o \mid X\}\}$$

Since $E\{\mathbf{e}_o \mid \mathbf{X}\} = 0$, from assumption 3, we have $E\{\mathbf{c}_{ls}\} = \mathbf{c}_o$. Thus c_{ls} is also unconditionally unbiased.

PROPERTY 7.2.8 The covariance matrix of c_{ls} corresponding to the error $c_{\mathrm{ls}} - c_o$ is

$$\boldsymbol{\Gamma}_{\mathrm{ls}} \triangleq E\{(c_{\mathrm{ls}}-c_o)(c_{\mathrm{ls}}-c_o)^{\mathrm{H}}\} = \sigma^2_{e_o}E\{(X^{\mathrm{H}}X)^{-1}\} \tag{7.2.51}$$

Proof. From (7.2.42), the conditional covariance matrix of c_{ls}, conditional on X, is

$$E\{(c_{\mathrm{ls}}-c_o)(c_{\mathrm{ls}}-c_o)^{\mathrm{H}} \mid X\} = \sigma^2_{e_o}(X^{\mathrm{H}}X)^{-1} \tag{7.2.52}$$

For the unconditional covariance, we use the decomposition of covariance rule to obtain

$$E\{(c_{\mathrm{ls}}-c_o)(c_{\mathrm{ls}}-c_o)^{\mathrm{H}}\} = E_X\{E\{(c_{\mathrm{ls}}-c_o)(c_{\mathrm{ls}}-c_o)^{\mathrm{H}} \mid X\}\}$$
$$+ E_X\{(E\{c_{\mathrm{ls}} \mid X\}-c_o)(E\{c_{\mathrm{ls}} \mid X\}-c_o)^{\mathrm{H}}\}$$

The second term on the right-hand side above is equal to zero since $E\{c_{\mathrm{ls}} \mid X\} = c_o$ and hence

$$E\{(\boldsymbol{c}_{\text{ls}}-\boldsymbol{c}_o)(\boldsymbol{c}_{\text{ls}}-\boldsymbol{c}_o)^{\text{H}}\} = E_X\{E\{(\boldsymbol{c}_{\text{ls}}-\boldsymbol{c}_o)(\boldsymbol{c}_{\text{ls}}-\boldsymbol{c}_o)^{\text{H}} \mid \boldsymbol{X}\}\}$$
$$= E_X\{\sigma_{e_o}^2(\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\} = \sigma_{e_o}^2 E\{(\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\}$$

Thus the earlier result in (7.2.42) is modified by the expected value (or averaging) of $(\mathbf{X}^H\mathbf{X})^{-1}$.

One important conclusion about the statistical properties of the LS estimator is that the results obtained for the deterministic data matrix $\boldsymbol{X}$ are also valid for the stochastic case. This conclusion also applies for the Markov estimators and maximum likelihood estimators (Greene 1993).

7.3 Least-Squares FIR Filters

We will now apply the theory of linear LS error estimation to the design of FIR filters. The treatment closely follows the notation and approach in Section 5.4. Recall that the filtering error is

$$e(n) = y(n) - \sum_{k=0}^{M-1} h(k)\,x(n-k) \triangleq y(n) - \boldsymbol{c}^{\text{H}}\boldsymbol{x}(n) \tag{7.3.1}$$

where $y(n)$ is the desired response,

$$\boldsymbol{x}(n) = [x(n)\; x\,(n-1)\; \cdots\; x(n-M+1)]^{\text{T}} \tag{7.3.2}$$

is the input data vector, and

$$\boldsymbol{c} = [c_0\; c_1\; \cdots\; c_{M-1}]^{\text{T}} \tag{7.3.3}$$

is the filter coefficient vector related to impulse response by $c_k = h^*(k)$. Suppose that we take measurements of the desired response $y(n)$ and the input signal $x(n)$ over the time interval $0 \le n \le N-1$. We hold the coefficients $\{c_k\}_0^{M-1}$ of the filter constant within this period and set any other required data samples equal to zero. For example, at time $n=0$, that is, when we take the first measurement $x(0)$, the filter needs the samples $x(0)$, $x(-1),\cdots,x(-M+1)$ to compute the output sample $\hat{y}(0)$. Since the samples $x(-1), \cdots, x(-M+1)$ are not available, to operate the filter, we should replace them with *arbitrary* values or start the filtering operation at time $n=M-1$. Indeed, for $M-1 \le n \le N-1$, all the input samples of $x(n)$ required by the filter to compute the output $\{\hat{y}(n)\}_{M-1}^{N-1}$ are available. If we want to compute the output while the last sample $x(N-1)$ is still in the filter memory, we must continue the filtering operation until $n = N+M-2$. Again, we need to assign arbitrary values to the unavailable samples $x(N),\cdots,x(N+M-2)$. Most often, we set the unavailable samples equal to zero, which can be thought of as windowing the sequences $x(n)$ and $y(n)$ with a rectangular window. To simplify the illustration, suppose that $N=7$ and $M=3$. Writing (7.3.1) for $n=0,\ 1,\ \cdots,\ N+M-1$ and arranging in matrix form, we obtain

$$\begin{array}{r} 0\rightarrow \\ \\ M-1\rightarrow \\ \\ \\ \\ N-1\rightarrow \\ \\ N+M-2\rightarrow \end{array}
\left[\begin{array}{c} e^*(0) \\ e^*(1) \\ \hline e^*(2) \\ e^*(3) \\ e^*(4) \\ e^*(5) \\ e^*(6) \\ \hline e^*(7) \\ e^*(8) \end{array}\right]
=
\left[\begin{array}{c} y^*(0) \\ y^*(1) \\ \hline y^*(2) \\ y^*(3) \\ y^*(4) \\ y^*(5) \\ y^*(6) \\ \hline 0 \\ 0 \end{array}\right]
-
\left[\begin{array}{ccc} x^*(0) & 0 & 0 \\ x^*(1) & x^*(0) & 0 \\ \hline x^*(2) & x^*(1) & x^*(0) \\ x^*(3) & x^*(2) & x^*(1) \\ x^*(4) & x^*(3) & x^*(2) \\ x^*(5) & x^*(4) & x^*(3) \\ x^*(6) & x^*(5) & x^*(4) \\ \hline 0 & x^*(6) & x^*(5) \\ 0 & 0 & x^*(6) \end{array}\right]
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} \tag{7.3.4}$$

or, in general,

$$\boldsymbol{e} = \mathbf{y} - \boldsymbol{X}\boldsymbol{c} \tag{7.3.5}$$

where the exact form of $\boldsymbol{e}$, $\mathbf{y}$, and $\mathbf{X}$ depends on the range $N_i \le n \le N_f$ of measurements to be used, which in turn determines the range of summation

$$E=\sum_{n=N_i}^{N_f}|e(n)|^2=\boldsymbol{e}^{\mathrm{H}}\boldsymbol{e} \tag{7.3.6}$$

in the LS criterion. The LS FIR filter is found by solving the LS normal equations

$$(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})\boldsymbol{c}_{\mathrm{ls}}=\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y} \tag{7.3.7}$$

or

$$\hat{\boldsymbol{R}}\boldsymbol{c}_{\mathrm{ls}}=\hat{\boldsymbol{d}} \tag{7.3.8}$$

with an LS error of

$$E_{\mathrm{ls}}=E_y-\hat{\mathbf{d}}^{\mathrm{H}}\mathbf{c}_{\mathrm{ls}} \tag{7.3.9}$$

where E_y is the energy of the desired response signal. The elements of the time-average correlation matrix **R** are given by

$$\hat{r}_{ij}=\tilde{\boldsymbol{x}}_i^{\mathrm{H}}\tilde{\boldsymbol{x}}_j=\sum_{n=N_i}^{N_f}x(n+1-i)x^*(n+1-j) \qquad 1\le i,\ j\le M \tag{7.3.10}$$

where $\tilde{\boldsymbol{x}}_i$ are the columns of data matrix $\boldsymbol{X}$. A simple manipulation of (7.3.10) leads to

$$\hat{r}_{i+1,j+1}=-\hat{r}_{ij}+x(N_i-i)x^*(N_i-j)-x(N_f+1-i)x^*(N_f+1-j) \quad 1\le i,j<M \tag{7.3.11}$$

which relates the elements of matrix $\boldsymbol{R}$ that are located on the same diagonal. This property holds because the columns of $\boldsymbol{X}$ are obtained by shifting the first column. The recursion in (7.3.11) suggests the following way of efficiently computing $\boldsymbol{R}$:

1. Compute the first row of $\boldsymbol{R}$ by using (7.3.10). This requires M dot products and a total of about $M(N_f-N_i)$ operations.
2. Compute the remaining elements in the upper triangular part of $\hat{\boldsymbol{R}}$, using (7.3.11). This required number of operations is proportional to M^2.
3. Compute the lower triangular part of $\boldsymbol{R}$, using the Hermitian symmetry relation $\hat{r}_{ji}=\hat{r}_{ij}^*$.

Notice that direct computation of the upper triangular part of $\boldsymbol{R}$ using (7.3.10), that is, without the recursion, requires approximately $M^2N/2$ operations, which increases significantly for moderate or large values of $\boldsymbol{M}$.

There are four ways to select the summation range $N_i\le n\le N_f$ that are used in LS filtering and prediction:

No windowing. If we set $N_i=M-1$ and $N_f=N-1$, we only use the available data and there are no distortions caused by forcing the data at the borders to artificial values.

Prewindowing. This corresponds to $N_i=0$ and $N_f=N-1$ and is equivalent to setting the samples $x(0)$, $x(-1)$, $\cdots$, $x(-M+1)$ equal to zero. As a result, the term $x(M-i)x(M-j)$ does not appear in (7.3.1). This method is widely used in LS adaptive filtering.

Postwindowing. This corresponds to $N_i=M-1$ and $N_f=N+M-2$ and is equivalent to setting the samples $x(N)$, $\cdots$, $x(N+M-2)$ equal to zero. As a result, the term $x(M-i)x(M-j)$ does not appear in (7.3.11). This method is not used very often for practical applications without prewindowing.

Full windowing. In this method, we impose both prewindowing and postwindowing (full windowing) to the input data and postwindowing to the desired response. The range of summation is from $N_i=0$ to $N_f=N+M-2$, and as a result of full windowing, Eq. (7.3.11) becomes $\hat{r}_{i+1,j+1}=\hat{r}_{ij}$. Therefore, the elements $\hat{r}_{ij}$, depend on $i-j$, and matrix $\boldsymbol{R}$ is Toeplitz. In this case, the normal equations (7.2.12) can be obtained from the Wiener-Hopf equations (5.4.11) by replacing the theoretical autocorrelations with their estimated values (see Section 4.2).

Clearly, as $N\gg M$ the performance difference between the various methods becomes insignificant. The no-windowing and full-windowing methods are known in the signal processing literature as the *autocorrelation* and

covariance methods, respectively (Makhoul 1975b). We avoid these terms because they can lead to misleading statistical interpretations. We notice that in the LS filtering problem, the data matrix $\boldsymbol{X}$ is Toeplitz and the normal equations matrix $\hat{\boldsymbol{R}} = \boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}$ is the product of two Toeplitz matrices. However, $\boldsymbol{R}$ is Toeplitz only in the full-windowing case when $\boldsymbol{X}$ is banded Toeplitz. In all other cases $\boldsymbol{R}$ is *near to Toeplitz* or $\boldsymbol{R}$ is *close to Toeplitz* in a sense made precise in Morf, et al. (1977).

The matrix $\boldsymbol{R}$ and vector $\boldsymbol{d}$, for the various windowing methods, are computed by using the MATLAB function `[R,d]=lsmatvec(x, M, method, y)`, which is based on (7.3.10) and (7.3.11). Then the LS filter is computed by `cls=R\d`. Figure 7.6 shows an FIR LSE filter operating in block processing mode.

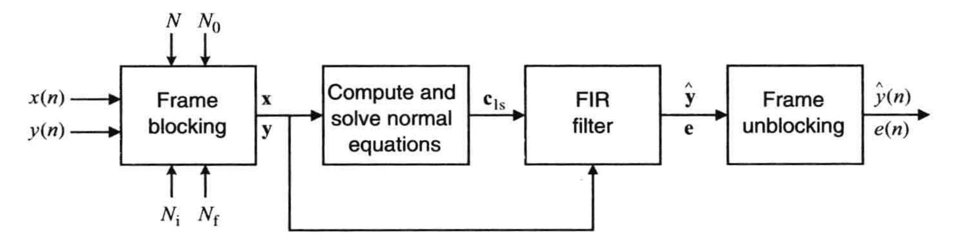


FIGURE 7.6
Block processing implementation of an FIR LSE filter.

EXAMPLE 7.3.1 To illustrate the design of least-squares FIR filters, suppose that we have a set of measurements of $x(n)$ and $y(n)$ for $0 \le n \le N-1$ with $N = 100$ that have been generated by the difference equation

$$y(n) = 0.5x(n) + 0.5x(n-1) + v(n)$$

The input $x(n)$ and the additive noise $v(n)$ are uncorrelated processes from a normal (Gaussian) distribution with mean $E\{x(n)\} = E\{v(n)\} = 0$ and variance $\sigma_x^2 = \sigma_v^2 = 1$. Fitting the model

$$\hat{y}(n) = h(0)x(n) + h(1)x(n-1)$$

to the measurements with the no-windowing LS criterion, we obtain

$$\boldsymbol{c}_{\mathrm{ls}} = \begin{bmatrix} 0.5361 \\ 0.5570 \end{bmatrix} \qquad \hat{\sigma}_e^2 = 1.0419 \qquad \hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1} = \begin{bmatrix} 0.0073 & -0.0005 \\ -0.0005 & 0.0071 \end{bmatrix}$$

using (7.3.7), (7.3.9), (7.2.44), and (7.2.42). If the mean of the additive noise is nonzero, for example, if $E\{v(n)\} = 1$, we get

$$\boldsymbol{c}_{\mathrm{ls}} = \begin{bmatrix} 0.4889 \\ 0.5258 \end{bmatrix} \qquad \hat{\sigma}_e^2 = 1.8655 \qquad \hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1} = \begin{bmatrix} 0.0131 & -0.0009 \\ -0.0009 & 0.0127 \end{bmatrix}$$

which shows that the variance of the estimates, that is, the diagonal elements of $\hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1}$, increases significantly. Suppose now that the recording device introduces an outlier in the input data at $x(30) = 20$. The estimated LS model and its associated statistics are given by

$$\boldsymbol{c}_{\mathrm{ls}} = \begin{bmatrix} 0.1796 \\ 0.1814 \end{bmatrix} \qquad \hat{\sigma}_e^2 = 1.6270 \qquad \hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1} = \begin{bmatrix} 0.0030 & 0.0000 \\ 0.0000 & 0.0030 \end{bmatrix}$$

Similarly, when an outlier is present in the output data, for example, at $y(30) = 20$, then the LS model and its statistics are

$$\boldsymbol{c}_{\mathrm{ls}} = \begin{bmatrix} 0.6303 \\ 0.4653 \end{bmatrix} \qquad \hat{\sigma}_e^2 = 5.0979 \qquad \hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1} = \begin{bmatrix} 0.0357 & -0.0025 \\ -0.0025 & 0.0347 \end{bmatrix}$$

In general, LS estimates are very sensitive to colored additive noise and outliers (Ljung 1987). Note that all the LS solutions in this example were produced with one sample realization $x(n)$ and that the results will vary for any other realizations.

LS inverse filters. Given a causal filter with impulse response $g(n)$, its inverse filter $h(n)$ is specified by $g(n) * h(n) = \delta(n - n_0)$, $n_0 \ge 0$. We focus on causal inverse filters, which are often infinite impulse response (IIR), and we wish to approximate them by some FIR filter $c_{\mathrm{ls}}(n) = h^*(n)$ that is optimum according to the LS criterion. In this case, the actual impulse response $g(n) * c_{\mathrm{ls}}^*(n)$ of the combined system deviates from the desired response $\delta(n - n_0)$, resulting in an error $e(n)$. The convolution equation

$$e(n) = \delta(n - n_0) - \sum_{k=0}^{M} c_{ls}^*(k)\, g(n-k) \tag{7.3.12}$$

can be formulated in matrix form as follows for $M = 2$ and $N = 6$

$$\begin{bmatrix} e^*(0) \\ e^*(1) \\ e^*(2) \\ e^*(3) \\ e^*(4) \\ e^*(5) \\ e^*(6) \\ e^*(7) \\ e^*(8) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} g^*(0) & 0 & 0 \\ g^*(1) & g^*(0) & 0 \\ g^*(2) & g^*(1) & g^*(0) \\ g^*(3) & g^*(2) & g^*(1) \\ g^*(4) & g^*(3) & g^*(2) \\ g^*(5) & g^*(4) & g^*(3) \\ g^*(6) & g^*(5) & g^*(4) \\ 0 & g^*(6) & g^*(5) \\ 0 & 0 & g^*(6) \end{bmatrix} \begin{bmatrix} c_{ls}(0) \\ c_{ls}(1) \\ c_{ls}(2) \end{bmatrix}$$

assuming that $n_0 = 0$. In general,

$$\boldsymbol{e} = \boldsymbol{\delta}_i - \boldsymbol{G}\boldsymbol{c}_{\text{ls}}^{(i)} \tag{7.3.13}$$

where $\boldsymbol{\delta}_i$ is a vector whose i th element is 1 and whose remaining elements are all zero. The LS inverse filter and the corresponding error are given by

$$(\boldsymbol{G}^{\text{H}}\boldsymbol{G})\boldsymbol{c}_{\text{ls}}^{(i)} = \boldsymbol{G}^{\text{H}}\boldsymbol{\delta}_i \tag{7.3.14}$$

and
$$E_{\text{ls}}^{(i)} = 1 - \boldsymbol{\delta}_i^{\text{H}}\boldsymbol{G}\boldsymbol{c}_{\text{ls}}^{(i)} = 1 - g^*(i)c_{\text{ls}}^{(i)}(i) \qquad 0 \le i \le M+N \tag{7.3.15}$$

Using the projection operators (7.2.29) and (7.2.30), we can express the LS error as

$$E_{\text{ls}}^{(i)} = \boldsymbol{\delta}_i^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})\boldsymbol{\delta}_i \tag{7.3.16}$$

where
$$\boldsymbol{P} = \boldsymbol{G}(\boldsymbol{G}^{\text{H}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\text{H}} \tag{7.3.17}$$

The total error for all possible deldya $0 \le i \le N+M$ can be written as

$$E_{\text{total}} = \sum_{i=0}^{N+M} E_{\text{ls}}^{(i)} = \text{tr}[\boldsymbol{D}^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})\boldsymbol{D}] \tag{7.3.18}$$

where
$$\boldsymbol{D} \triangleq [\boldsymbol{\delta}_0\ \boldsymbol{\delta}_1\ \boldsymbol{\delta}_2\ \cdots\ \boldsymbol{\delta}_{N+M}] = \boldsymbol{I} \tag{7.3.19}$$

is the $[(N+M+1)\times(N+M+1)$ identity matrix. Since $\boldsymbol{D} = \boldsymbol{I}$, $\boldsymbol{P} = \boldsymbol{P}^{\text{H}}$, and $\boldsymbol{P}^2 = \boldsymbol{P}$, we obtain

$$E_{\text{total}} = \text{tr}[\boldsymbol{D}^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})^{\text{H}}(\boldsymbol{P}-\boldsymbol{I})\boldsymbol{D}] = \text{tr}(\boldsymbol{I}-\boldsymbol{P}) = \text{tr}(\boldsymbol{I}) - \text{tr}(\boldsymbol{P})$$

or
$$E_{\text{total}} = N \tag{7.3.20}$$

because $\text{tr}(\mathbf{I}) = N+M+1$ and

$$\text{tr}(\boldsymbol{P}) = \text{tr}[\boldsymbol{G}(\boldsymbol{G}^{\text{H}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\text{H}}] = \text{tr}[\boldsymbol{G}^{\text{H}}\boldsymbol{G}(\boldsymbol{G}^{\text{H}}\boldsymbol{G})^{-1}] = M+1$$

Hence, E_{total} depends on the length $N+1$ of the filter $g(n)$ and is independent of the length $M+1$ of the inverse filter $c_{\text{ls}}(n)$. If the minimum $E_{ls}^{(i)}$, for a given N, occurs at delay $i = i_0$, we have

$$E_{\text{ls}}^{(i_0)} \le \frac{N}{N+M+1} \tag{7.3.21}$$

which shows that $E_{\text{ls}}^{(i_0)} \to 0$ as $M \to \infty$ (Claerbout and Robinson 1963).

EXAMPLE 7.3.2 Suppose that $g(n) = \delta(n) - \alpha\delta(n-1)$, where α is a real constant. The exact inverse filter is

$$H(z) = \frac{1}{1-\alpha z^{-1}} - h(n) = \alpha^n u(n)$$

and is minimum-phase only if $-1 < \alpha < 1$. The inverse LS filter for $M = 1$ and $N \ge 2$ is obtained by applying (7.3.14) with

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \\ 0 & -\alpha \end{bmatrix} \quad \text{and} \quad \boldsymbol{\delta} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The normal equations are

$$\begin{bmatrix} 1+\alpha^2 & -\alpha \\ -\alpha & 1+\alpha^2 \end{bmatrix} \begin{bmatrix} c_{ls}(0) \\ c_{ls}(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{7.3.22}$$

leading to the LS inverse filter

$$c_{\text{ls}}(0) = \frac{1+\alpha^2}{1+\alpha^2+\alpha^4} \qquad c_{\text{ls}}(1) = \frac{\alpha}{1+\alpha^2+\alpha^4}$$

with LS error

$$E_{\text{ls}} = 1 - c_{\text{ls}}(0) = \frac{\alpha^4}{1+\alpha^2+\alpha^4}$$

The system function of the LS inverse filter is

$$H_{\text{ls}}(z) = \frac{1+\alpha^2}{1+\alpha^2+\alpha^4}\left(1+\frac{\alpha}{1+\alpha^2}z^{-1}\right)$$

and has a zero at $z_1 = -\alpha/(1+\alpha^2) = -1/(\alpha+\alpha^{-1})$. Since $|z_1|<1$ for any value of α, the LS inverse filter is minimum-phase even if $g(n)$ is not. This stems from the fact that the normal equations (7.3.22) specify a one-step forward linear predictor with a correlation matrix that is Toeplitz and positive definite for any value of α (see Section 6.4).

7.4 Linear Least-Squares Signal Estimation

We now discuss the application of the LS method to general signal estimation, FLP, BLP, and combined forward and backward linear prediction. The reader is advised to review Section 5.5, which provides a detailed discussion of the same problems for the MMSE criterion. The presentation in this section closely follows the viewpoint and notation in Section 5.5.

7.4.1 Signal Estimation and Linear Prediction

Suppose that we wish to compute the linear LS signal estimator $c_k^{(i)}$ defined by

$$e^{(i)}(n) = \sum_{k=0}^{M} c_k^{(i)*} x(n-k) = \boldsymbol{c}^{(i)H}\bar{\boldsymbol{x}}(n) \qquad \text{with} c_i^{(i)} \triangleq 1 \tag{7.4.1}$$

from the data $x(n)$, $0 \le n \le N-1$. Using (7.4.1) and following the process that led to (7.3.4), we obtain

$$\boldsymbol{e}^{(i)} = \bar{\boldsymbol{X}}\boldsymbol{c}^{(i)} \tag{7.4.2}$$

where

$$\bar{\boldsymbol{X}} = \begin{bmatrix} x^*(0) & 0 & \cdots & 0 \\ x^*(1) & x^*(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x^*(M) & x^*(M-1) & \cdots & x^*(0) \\ \vdots & \vdots & & \vdots \\ x^*(N-1) & x^*(N-2) & \cdots & x^*(N-M-1) \\ 0 & x^*(N-1) & \cdots & x^*(N-M) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x^*(N-1) \end{bmatrix} \tag{7.4.3}$$

is the combined data and desired response matrix with all the unavailable samples set equal to zero (full windowing). Matrix $\bar{\boldsymbol{X}}$ can be partitioned columnwise as

$$\bar{\boldsymbol{X}} = [\boldsymbol{X}_1 \;\; \boldsymbol{y} \;\; \boldsymbol{X}_2] \tag{7.4.4}$$

where **y**, the desired response, is the ith column of $\bar{X}$. Using (7.4.4), we can easily show that the LS signal estimator $c_{ls}^{(i)}$ and the associated LS error $E_{ls}^{(i)}$ are determined by

$$(\bar{X}^{H}\bar{X})c_{ls}^{(i)} = \begin{bmatrix} 0 \\ E_{ls}^{(i)} \\ 0 \end{bmatrix} \tag{7.4.5}$$

where $E_{ls}^{(i)}$ is the ith element of the right-hand side vector (see Problem 7.3). If we define the time-average correlation matrix

$$\bar{R} \triangleq \bar{X}^{H}\bar{X} \tag{7.4.6}$$

and use the augmented normal equations in (7.4.5), we obtain a set of equations that have the same form as (5.5.12), the equations for the MMSE signal estimator. Therefore, after we have computed $\bar{R}$, using the command `Rbar=lsmatvec(x, M+1, method)`, we can use the steps in Table 5.3 to compute the LS forward linear predictor (FLP), the backward linear predictor (BLP), the symmetric smoother, or any other signal estimator with delay i. Again, we use the standard notation $E_{ls}^{(0)} = E^{f}$ and $c_{ls}^{(0)} = a$ for the FLP and $E_{ls}^{(M)} = E^{b}$ and $c_{ls}^{(M)} = b$ for the BLP.

All formulas given in Section 5.5 hold for LS signal estimators if the matrix $R(n)$ is replaced by $\bar{R}$. However, we stress that although the optimum MMSE signal estimator $c_{o}^{(i)}(n)$ is a deterministic vector, the LS signal estimator $c_{ls}^{(i)}$ is a random vector that is a function of the random measurements $x(n)$, $0 \le n \le N-1$. In the full-windowing case, matrix $\bar{R}$ is Toeplitz; if it is also positive definite, then the FLP is minimum-phase. Although the use of full windowing leads to these nice properties, it also creates some "edge effects" and bias in the estimates because we try to estimate some signal values using values that are not part of the signal by forcing the samples leading and lagging the available data measurements to zero.

EXAMPLE 7.4.1. Suppose that we are given the signal segment $x(n) = \alpha^{n}$, $0 \le n \le N$, where α is an arbitrary complex-valued constant. Determine the first-order one-step forward linear predictor, using the full-windowing and no-windowing methods.

Solution. We start by forming the combined desired response and data matrix

$$\bar{X}^{H} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N) & 0 \\ 0 & x(0) & \cdots & x(N-1) & x(N) \end{bmatrix}$$

For the full-windowing method, the matrix

$$\bar{R} = \bar{X}^{H}\bar{X} = \begin{bmatrix} \hat{r}_x(0) & \hat{r}_x(1) \\ \hat{r}_x^{*}(1) & \hat{r}_x(0) \end{bmatrix}$$

is Toeplitz with elements

$$\hat{r}_x(0) = \sum_{n=0}^{N} |x(n)|^2 = \sum_{n=0}^{N} |\alpha|^{2n} = \frac{1-|\alpha|^{2(N+1)}}{1-|\alpha|^2}$$

and

$$\hat{r}_x(1) = \sum_{n=1}^{N} x(n)x^{*}(n-1) = \sum_{n=1}^{N} \alpha^{n}(\alpha^{*})^{n-1} = \alpha^{*}\frac{1-|\alpha|^{2N}}{1-|\alpha|^2}$$

Therefore, we have

$$\begin{bmatrix} \hat{r}_x(0) & \hat{r}_x(1) \\ \hat{r}_x^{*}(1) & \hat{r}_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} E_1^{f} \\ 0 \end{bmatrix}$$

whose solution gives

$$a_1^{(1)} = -\frac{\hat{r}_x^{*}(1)}{\hat{r}_x(0)} = -\alpha\frac{1-|\alpha|^{2N}}{1-|\alpha|^{2(N+1)}}$$

and

$$E_1^{f} = \hat{r}_x(0) + \hat{r}_x(1)a_1^{(1)} = \frac{1-|\alpha|^{2(2N+1)}}{1-|\alpha|^{2(N+1)}}$$

Since for every sequence $|\hat{r}_x(l)| \le |\hat{r}_x(0)|$, we have $|a_1^{(1)}| \le 1$; that is, the obtained prediction error filter always is minimum-phase. Furthermore, if $|\alpha| < 1$, then $\lim_{N\to\infty} a_1^{(1)} = -\alpha$ and $\lim_{N\to\infty} E_1^{f} = 1 = x(0)$. In the no-windowing case, the matrix

$$\bar{\boldsymbol{R}} = \bar{\boldsymbol{X}}^{\mathrm{H}}\bar{\boldsymbol{X}} = \begin{bmatrix} \hat{r}_{11} & \hat{r}_{12} \\ \hat{r}_{12}^* & \hat{r}_{22} \end{bmatrix}$$

is Hermitian but not Toeplitz with elements

$$\hat{r}_{11} = \sum_{n=1}^{N} |x(n)|^2 = |\alpha|^2 \frac{1-|\alpha|^{2N}}{1-|\alpha|^2} \qquad \hat{r}_{22} = \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1-|\alpha|^{2N}}{1-|\alpha|^2}$$

$$\hat{r}_{12} = \sum_{n=1}^{N} x(n)\, x^*(n-1) = \alpha^* \frac{1-|\alpha|^{2N}}{1-|\alpha|^2}$$

Solving the linear system

$$\begin{bmatrix} \hat{r}_{11} & \hat{r}_{12} \\ \hat{r}_{12}^* & \hat{r}_{22} \end{bmatrix} \begin{bmatrix} 1 \\ \bar{a}_1^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{E}_1^f \\ 0 \end{bmatrix}$$

we obtain

$$\bar{a}_1^{(1)} = -\frac{\hat{r}_{12}^*}{\hat{r}_{22}} = -\alpha$$

and

$$\bar{E}_1^{\mathrm{f}} = \hat{r}_{11} + \hat{r}_{12}\bar{a}_1^{(1)} = 0$$

We see that the no-windowsing method provides a perfect linear predictor because there is no distortion due to windowing. However, the obtained prediction error filter is minimum-phase only when $|\alpha| < 1$.

EXAMPLE 7.4.2 To illustrate the statistical properties of least-squares FLP, we generate $K = 500$ realizations of the MA(1) process $x(n) = \omega(n) + \frac{1}{2}\omega(n-1)$, where $\omega(n) \sim \mathrm{WN}(0,1)$ (see Example 5.5.2). Each realization $x(\zeta_i, n)$ has duration $N = 100$ samples. We use these data to design an $M = 2$ order FLP, using the no-windowing LS method. The estimated mean and variance of the obtained K FLP vectors are

$$\mathrm{Mean}\{\boldsymbol{a}(\zeta_i)\} = \begin{bmatrix} -0.4695 \\ 0.1889 \end{bmatrix} \quad \text{and} \quad \mathrm{var}\{\boldsymbol{a}(\zeta_i)\} = \begin{bmatrix} 0.0086 \\ 0.0092 \end{bmatrix}$$

whereas the average of the variances $\hat{\sigma}_e^2$ is 0.9848. We notice that both means are close to the theoretical values obtained in Example 5.5.2. The covariance matrix of a given LS estimate $\boldsymbol{a}_{ls}$ was found to be

$$\hat{\sigma}_e^2 \hat{\boldsymbol{R}}^{-1} = \begin{bmatrix} 0.0099 & -0.0043 \\ -0.0043 & 0.0099 \end{bmatrix}$$

whose diagonal elements are close to the components of $\mathrm{var}\{\boldsymbol{a}\}$, as expected. The bias in the estimate $\boldsymbol{a}_{\mathrm{ls}}$ results from the fact that the residuals in the LS equations are correlated with each other (see Problem 7.4).

7.4.2 Combined Forward and Backward Linear Prediction (FBLP)

For stationary stochastic processes, the optimum MMSE forward and backward linear predictors have even conjugate symmetry, that is,

$$\boldsymbol{a}_o = \boldsymbol{J}\boldsymbol{b}_o^* \tag{7.4.7}$$

because both directions of time have the same second-order statistics. Formally, this property stems from the Toeplitz structure of the autocorrelation matrix (see Section 5.5). However, we could possibly improve performance by minimizing the total forward and backward squared error

$$E^{\mathrm{fb}} = \sum_{n=N_{\mathrm{i}}}^{N_{\mathrm{f}}} \{|e^{\mathrm{f}}(n)|^2 + |e^{\mathrm{b}}(n)|^2\} = (\boldsymbol{e}^{\mathrm{f}})^H \boldsymbol{e}^{\mathrm{f}} + (\boldsymbol{e}^{\mathrm{b}})^H \boldsymbol{e}^{\mathrm{b}} \tag{7.4.8}$$

under the constraint

$$\boldsymbol{a}^{\text{fb}} \triangleq \boldsymbol{a} = \boldsymbol{J}\boldsymbol{b}^* \tag{7.4.9}$$

The FLP and BLP overdetermined sets of equations are

$$\boldsymbol{e}^{\text{f}} = \bar{\boldsymbol{X}}\begin{bmatrix} 1 \\ \boldsymbol{a} \end{bmatrix} \quad \text{and} \quad \boldsymbol{e}^{\text{b}} = \bar{\boldsymbol{X}}\begin{bmatrix} \boldsymbol{b} \\ 1 \end{bmatrix} \tag{7.4.10}$$

or

$$\text{e}^{\text{f}} = \bar{\boldsymbol{X}}\begin{bmatrix} 1 \\ \boldsymbol{a}^{\text{fb}} \end{bmatrix} \quad \text{and} \quad \boldsymbol{e}^{\text{b}*} = \bar{\boldsymbol{X}}^*\begin{bmatrix} \boldsymbol{b}^* \\ 1 \end{bmatrix} = \bar{\boldsymbol{X}}^*\boldsymbol{J}\begin{bmatrix} 1 \\ \boldsymbol{a}^{\text{fb}} \end{bmatrix} \tag{7.4.11}$$

where we have used (7.4.9) and the property $\boldsymbol{J}\boldsymbol{J} = \boldsymbol{I}$ of the exchange matrix. If we combine the above two equations as

$$\begin{bmatrix} \boldsymbol{e}^{\text{f}} \\ \boldsymbol{e}^{\text{b}^*} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^*\boldsymbol{J} \end{bmatrix}\begin{bmatrix} 1 \\ \boldsymbol{a}^{\text{fb}} \end{bmatrix} \tag{7.4.12}$$

then the forward-backward linear predictor that minimizes E^{fb} is given by (see Problem 7.5)

$$\begin{bmatrix} \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^*\boldsymbol{J} \end{bmatrix}^{\text{H}}\begin{bmatrix} \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^*\boldsymbol{J} \end{bmatrix}\begin{bmatrix} 1 \\ \boldsymbol{a}_{\text{ls}}^{\text{fb}} \end{bmatrix} = \begin{bmatrix} E_{\text{ls}}^{\text{fb}} \\ 0 \end{bmatrix}$$

or

$$\bar{\boldsymbol{X}}^{\text{H}}\bar{\boldsymbol{X}} + \boldsymbol{J}\bar{\boldsymbol{X}}^{\text{T}}\bar{\boldsymbol{X}}^*\boldsymbol{J})\begin{bmatrix} 1 \\ \boldsymbol{a}_{\text{ls}}^{\text{fb}} \end{bmatrix} = \begin{bmatrix} E_{\text{ls}}^{\text{fb}} \\ 0 \end{bmatrix} \tag{7.4.13}$$

which can be solved by using the steps described in Table 5.3. The time-average forward-backward correlation matrix

$$\hat{\boldsymbol{R}}_{\text{fb}} \triangleq \bar{\boldsymbol{X}}^{\text{H}}\bar{\boldsymbol{X}} + \boldsymbol{J}\bar{\boldsymbol{X}}^{\text{T}}\bar{\boldsymbol{X}}^*\boldsymbol{J} \tag{7.4.14}$$

with elements

$$\hat{r}_{ij}^{\text{fb}} = \hat{r}_{ij} + \hat{r}_{M-i,M-j}^* \qquad 0 \le i, j \le M \tag{7.4.15}$$

is persymmetric; that is, $\boldsymbol{J}\hat{\boldsymbol{R}}_{\text{fb}}\boldsymbol{J} = \hat{\boldsymbol{R}}_{\text{fb}}^*$ and its elements are conjugate symmetric about both main diagonals. In MATLAB we compute $\hat{\boldsymbol{R}}_{\text{fb}}$ by these commands:

```
Rbar=lsmatvec(x, M+1, method)
Rfb=Rbar+flipud(fiplr(conj(Rbar)))
```

The FBLP method is used with *no windowing* and was originally introduced independently by Ulrych and Clayton (1976) and Nuttall (1976) as a spectral estimation technique under the name *modified covariance method* (see Section 8.2). If we use full windowing, then $\boldsymbol{a}^{\text{fb}} = (\boldsymbol{a} + \boldsymbol{J}\boldsymbol{b}^*)/2$ (see Problem 7.6).

7.4.3 Narrowband Interference Cancelation

Several practical applications require the removal of *narrowband interference* (*NBI*) from a wideband desired signal corrupted by additive white noise. For example, ground and foliage-penetrating radars operate from 0.01 to 1 GHz and use either an impulse or a chirp waveform. To achieve high resolution, these waveforms are extremely wideband, occupying at least 100 MHz within the range of 0.01 to 1 GHz. However, these frequency ranges are extensively used by TV and FM stations, cellular phones, and other relatively narrowband (less than 1 MHz) radio-frequency (RF) sources. Clearly, these sources spoil the radar returns with narrowband RF interference (Miller et al. 1997). Since the additive noise is often due to the sensor circuitry, it will be referred to as *sensor thermal noise*. Next we provide a practical solution to this problem, using an LS linear predictor. Suppose that the corrupted signal $x(n)$ is given by

$$x(n) = s(n) + y(n) + v(n) \tag{7.4.16}$$

where

$$s(n) = \text{signal of interest} \tag{7.4.17}$$

$$y(n) = \text{narrowband interference}$$

$$v(n) = \text{thermal (white) noise}$$

are the individual components, assumed to be stationary stochastic processes.

We wish to design an NBI canceler that estimates and rejects the interference signal $y(n)$ from the signal $x(n)$, while preserving the signal of interest $s(n)$. Since signals $y(n)$ and $x(n)$ are correlated, we can form an estimate of the NBI using the optimum linear estimator

$$\hat{y}(n) = \boldsymbol{c}_o^{\mathrm{H}} \boldsymbol{x}(n-D) \tag{7.4.18}$$

where

$$\boldsymbol{R}\boldsymbol{c}_o = \boldsymbol{d} \tag{7.4.19}$$

$$\boldsymbol{R} = E\{\boldsymbol{x}(n-D)\boldsymbol{x}^{\mathrm{H}}(n-D)\} \tag{7.4.20}$$

$$\boldsymbol{d} = E\{\boldsymbol{x}(n-D)\, y^*(n)\} \tag{7.4.21}$$

and D is an integer delay whose use will be justified shortly. Note that if $D=1$, then (7.4.18) is the LS forward linear predictor. If $\hat{y}(n) = y(n)$, the output of the canceler is $x(n) - \hat{y}(n) = s(n) + v(n)$; that is, the NBI is completely excised, and the desired signal is corrupted by white noise only and is said to be thermal noise–limited.

Since, in practice, the required second-order moments are not available, we need to use an LS estimator instead. However, the quantity $\boldsymbol{X}^H \boldsymbol{y}$ in (7.2.21) requires the NBI signal $y(n)$, which is also not available. To overcome this obstacle, consider the optimum MMSE D-step forward linear predictor

$$e^f(n) = x(n) + \boldsymbol{a}^{\mathrm{H}}\, \boldsymbol{x}(n-D) \tag{7.4.22}$$

$$\boldsymbol{R}\boldsymbol{a} = -\boldsymbol{r}^f \tag{7.4.23}$$

where $\boldsymbol{R}$ is given by (7.4.20) and

$$\boldsymbol{r}^f = E\{\boldsymbol{x}(n-D)x^*(n)\} \tag{7.4.24}$$

In many NBI cancelation applications, the components of the observed signal have the following properties:

1. The desired signal $s(n)$, the NBI $y(n)$, and the thermal noise $v(n)$ are mutually uncorrelated.
2. The thermal noise $v(n)$ is white; that is, $r_v(l) = \sigma_v^2 \delta(l)$.
3. The desired signal $s(n)$ is wideband and therefore has a short correlation length; that is, $r_v(l) = 0$ for $|l| \geq D$.
4. The NBI has a long correlation length; that is, its autocorrelation takes significant values over the range $0 \leq |l| \leq M$ for $M > D$.

In practice, the second and third properties mean that the desired signal and the thermal noise are approximately uncorrelated after a certain small lag. These are precisely the properties exploited by the canceler to separate the NBI from the desired signal and the background noise.

As a result of the first assumption, we have

$$E\{x(n-k)y^*(n)\} = E\{y(n-k)y^*(n)\} = r_y(k) \qquad \text{for all } k \tag{7.4.25}$$

and

$$r_x(l) = r_s(l) + r_y(l) + r_v(l) \tag{7.4.26}$$

Making use of the second and third assumptions, we have

$$r_x(l) = r_y(l) \qquad \text{for} \quad l \neq 0,\ 1,\ \cdots,\ D-1 \tag{7.4.27}$$

The exclusion of the lags for $1 \neq 0,\ 1,\ \cdots,\ D-1$ in $\boldsymbol{r}$ and $\boldsymbol{d}$ is critical, and we have arranged for that by forcing the filter and the predictor to form their estimates using the *delayed* data vector $\boldsymbol{x}(n-D)$. From (7.4.21), (7.4.24), and (7.4.27), we conclude that $\boldsymbol{d} = \boldsymbol{r}^{\mathrm{f}}$ and therefore $\boldsymbol{c}_o = \boldsymbol{a}_o$. Thus, the optimum NBI estimator $\boldsymbol{c}_o$ is equal to the

D-step linear predictor $\boldsymbol{a}_o$, which can be determined exclusively from the input signal $x(n)$. The cleaned signal is

$$x(n) - \hat{y}(n) = x(n) + \boldsymbol{a}_o^{\mathrm{H}} \boldsymbol{x}(n-D) = e^{\mathrm{f}}(n) \tag{7.4.28}$$

which is identical to the D-step forward prediction error. This leads to the linear prediction NBI canceler shown in Figure 7.7.

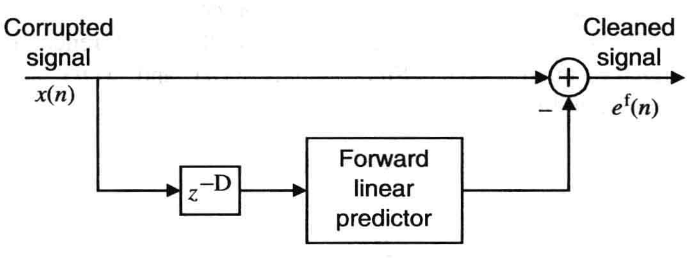


FIGURE 7.7
Block diagram of linear prediction NBI canceler.

To illustrate the performance of the linear prediction NBI canceler, we consider an impulse radar operating in a location with commercial radio and TV stations. The desired signal is a short-duration impulse corrupted by additive thermal noise and NBI (see Figure 7.8). The spectrum of the NBI is shown in Figure 7.9. We use a block of data $(N = 4096)$ to design an FBLP with $D = 1$ and $M = 100$ coefficients, using the LS criterion with no windowing. Then we compute the cleaned signal, using (7.4.28). The cleaned signal, its spectrum, and the magnitude response of the NBI canceler are shown in Figures 7.8 and 7.9. We see that the canceler acts as a notch filter that optimally puts notches at the peaks of the NBI. A detailed description of the design of optimum least-squares NBI cancelers is given in Problem 7.27.

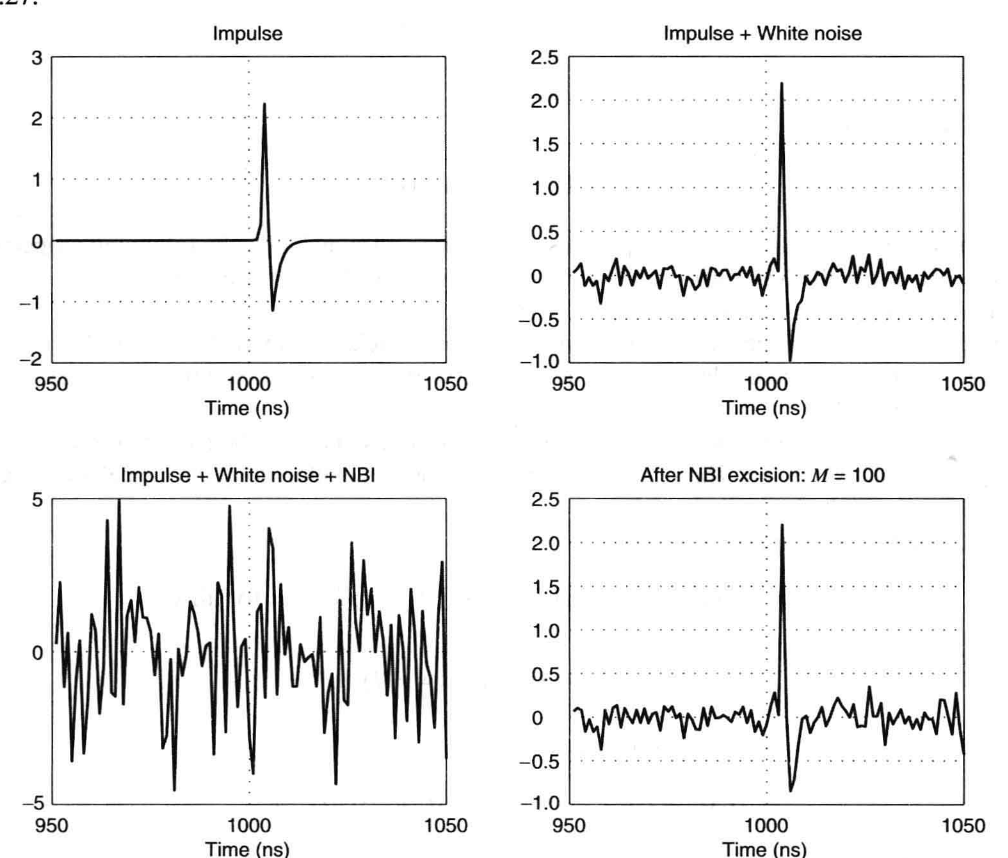


FIGURE 7.8
NBI cancelation: time-domain results.

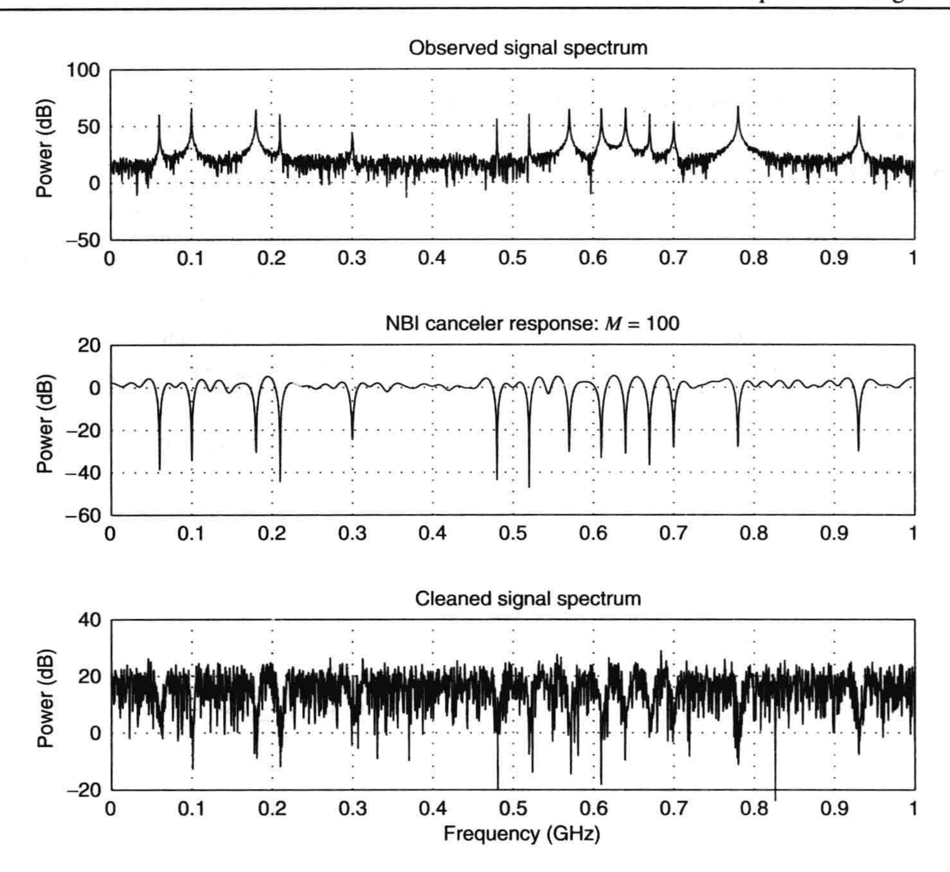


FIGURE 7.9
NBI cancelation: frequency-domain results.

7.5 LS Computations Using the Normal Equations

The solution of the normal equations for both MMSE and LSE estimation problems is computed by using the same algorithms. The key difference is that in MMSE estimation $\boldsymbol{R}$ and $\boldsymbol{d}$ are known, whereas in LSE estimation they need to be computed from the observed input and desired response signal samples. Therefore, it is natural to want to take advantage of the same algorithms developed for MMSE estimation in Chapter 6, whenever possible. However, keep in mind that despite algorithmic similarities, there are fundamental differences between the two classes of estimators that are dictated by the different nature of the the criteria of performance (see Section 8.1). In this section, we show how the computational algorithms and structures developed for linear MMSE estimation can be applied to linear LSE estimation, relying heavily on the material presented in Chapter 6.

7.5.1 Linear LSE Estimation

The computation of a general linear LSE estimator requires the solution of a linear system

$$\hat{\boldsymbol{R}}\boldsymbol{c}_{\text{ls}} = \hat{\boldsymbol{d}} \tag{7.5.1}$$

where the time-average correlation matrix $\boldsymbol{R}$ is Hermitian and positive definite [see (7.2.25)]. We can solve (7.5.1) by using the LDL^H or the Cholesky decomposition introduced in Section 5.3. The computation of linear LSE estimators involves the steps summarized in Table 7.1. We again stress that the major computational effort is involved in the computation of $\boldsymbol{R}$ and $\boldsymbol{d}$.

TABLE 7.1

Comparison between the LDL^H and Cholesky decomposition methods for the solution of normal equations.

Step	LDL^H decomposition	Cholesky decomposition	Description
1	$\hat{R} = X^H X, \hat{d} = X^H y$		Normal equations $\hat{R}c_{ls} = \hat{d}$
2	$\hat{R} = LDL^H$	$\hat{R} = \mathcal{L}\mathcal{L}^H$	Triangular decomposition
3	$LDk = \hat{d}$	$\mathcal{L}\tilde{k} = \hat{d}$	Forward substitution $\rightarrow k$ or k
4	$L^H c_{ls} = k$	$\mathcal{L}^H c_{ls} = \tilde{k}$	Backward substitution $\rightarrow c_{ls}$
5	$E_{ls} = E_y - k^H D k$	$E_{ls} = E_y - \tilde{k}^H \tilde{k}$	LSE computation
6	$e_{ls} = y - Xc_{ls}$	$e_{ls} = y - Xc_{ls}$	Computation of residuals

Steps 2 and 3 in (5.3.16) can be facilitated by a single extended LDL^H decomposition. To this end, we form the augmented data matrix

$$\bar{X} = [X \; y] \tag{7.5.2}$$

and compute its time-average correlation matrix

$$\bar{R} = \bar{X}^H \bar{X} = \begin{bmatrix} X^H X & X^H y \\ y^H X & y^H y \end{bmatrix} = \begin{bmatrix} \hat{R} & \hat{d} \\ \hat{d}^H & E_y \end{bmatrix} \tag{7.5.3}$$

We then can show (see Problem 7.9) that the LDL^H decomposition of $\bar{\mathbf{R}}$ is given by

$$\bar{R} = \begin{bmatrix} L & 0 \\ k^H & 1 \end{bmatrix} \begin{bmatrix} D & 0 \\ 0^H & E_{ls} \end{bmatrix} \begin{bmatrix} L^H & k^H \\ 0^H & 1 \end{bmatrix} \tag{7.5.4}$$

and thus provides the vector k and the LSE E_{ls}. Therefore, we can solve the normal equations (7.5.1), using the LDL^H decomposition of $\bar{R}$ to compute L and k and then solving $L^H c_{ls} = k$ to compute $\mathbf{c}_{ls}$.

A careful inpection of the design equations for the general, mth-order, MMSE and LSE estimators, derived in Chapter 5 and summarized in Table 7.2, shows that the LSE equations can be obtained from the MMSE equations by replacing the linear operator $E\{\cdot\}$ by the linear operator $\sum_n (\cdot)$. As a result, all algorithms developed in Sections 6.1 and 6.2 can be used for linear LSE estimation problems.

TABLE 7.2

Comparison between the MMSE and LSE normal equations for general linear estimation.

	MMSE	LSE
Available information	$R_m(n),\; d_m(n)$	$\{\mathbf{x}_m(n), y(n), n_i \le n \le n_f\}$
Normal equations	$R_m(n)c_m(n) = d_m(n)$	$\hat{R}_m c_m = \hat{d}_m$
Minimum error	$P_m(n) = P_y(n) - d_m^H(n)c_m(n)$	$E_m = E_y - \hat{d}_m^H c_m$
Correlation matrix	$R_m(n) \triangleq E\{X_m(n)X_m^H(n)\}$	$\hat{R}_m = X_m^H X_m = \sum_{n=0}^{N-1} X_m(n)X_m^H(n)$
Cross-correlation vector	$d_m(n) \triangleq E\{X_m(n)y^*(n)\}$	$\hat{d}_m = X_m^H y = \sum_{n=0}^{N-1} X_m(n)y^*(n)$
Power	$P_y(n) = E\{\|y(n)\|^2\}$	$E_y = y^H y = \sum_{n=0}^{N-1} \|y(n)\|^2$

For example, we can easily see that $\hat{R}_M$, $\hat{d}_M$, L_M, D_M, and $\mathbf{k}_M$ have the optimum nesting property described in Section 6.1.1, that is, $\hat{R}_m = \hat{R}_M^{[m]}$ and so on. As a result, the factors of the ***LDL^H*** decomposition have the optimum nesting property, and we can obtain an order-recursive structure for the computation of the LSE estimate $\hat{y}_m(n)$. Indeed, if we define

$$\mathbf{w}_m(n) = \mathbf{L}_m^{-1}\mathbf{x}_m(n) \qquad 0 \le n \le N-1 \tag{7.5.5}$$

then
$$\hat{R}_m = \sum_{n=0}^{N-1} x_m(n) x_m^{\mathrm{H}}(n) = L_m \left[\sum_{n=0}^{N-1} w_m(n) w_m^{\mathrm{H}}(n) \right] L_m \triangleq L_m D_m L_m^{\mathrm{H}} \tag{7.5.6}$$

where the matrix D_m is diagonal because the LDL^H decomposition is unique. If we define the record vectors

$$\tilde{w}_j \triangleq [w_j(0)\ w_j(1)\ \cdots\ w_j(N-1)]^{\mathrm{H}} \tag{7.5.7}$$

and the data matrix

$$W_m \triangleq [\tilde{w}_1\ \tilde{w}_2\ \cdots\ \tilde{w}_m] \tag{7.5.8}$$

then
$$D_m = W_m^{\mathrm{H}} W_m = \operatorname{diag}\{\xi_1,\ \xi_2,\ \cdots,\ \xi_m\} \tag{7.5.9}$$

where
$$\xi_i = \sum_{n=0}^{N-1} |w_i(n)|^2 = \tilde{w}_i^{\mathrm{H}} \tilde{w}_i \tag{7.5.10}$$

From (7.5.9) we have

$$\tilde{w}_i^H \tilde{w}_j = 0 \qquad \text{for } i \neq j \tag{7.5.11}$$

that is, the columns of W_m are orthogonal and, in this sense, are the innovation vectors of the columns of data matrix X_m, according to the LS interpretation of orthogonality introduced in Section 7.2.

Following the approach in Section 61.5, we can show that the following order-recursive algorithm

$$\begin{aligned} w_m(n) &= x_m(n) - \sum_{i=1}^{m-1} l_{i-1}^{(m-1)*} w_i(n) \\ \hat{y}_m(n) &= \hat{y}_{m-1}(n) + k_m^* w_m(n) \end{aligned} \tag{7.5.12}$$

or
$$e_m(n) = e_{m-1}(n) - k_m^* \omega_m(n)$$

computed for $n = 0,\ 1,\ \ldots,\ $ N-1 and $m = 1,\ 2,\ \ldots,\ $ M, provides the LSE estimates for orders $1 \le m \le M$.

The statistical interpretations of innovation and partial correlation for $\omega_m(n)$ and k_{m+1} hold now in a deterministic LSE sense. For example, the *partial correlation* between $\tilde{Y}$ and $\tilde{X}_{m+1}$ is defined by using the residual records $\tilde{e}_m = \tilde{y} - X_m c_m$ and $\tilde{e}_m^b = \tilde{x}_{m+1} + X_m b_m$, where b_m is the least-squares error BLP. Indeed, if $\beta_{m+1} \triangleq \tilde{e}_m^H \tilde{e}_m^b$, we can show that $k_{m+1} = \beta_{m+1}/\xi_{m+1}$ (see Problem 7.11).

EXAMPLE 7.5.1 Solve the LS problem with the following data matrix and desired response signal:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

Solution. We start by computing the time-average correlation matrix and cross-correlation vector

$$\hat{R} = \begin{bmatrix} 15 & 8 & 13 \\ 8 & 6 & 6 \\ 13 & 6 & 12 \end{bmatrix} \qquad \hat{d} = \begin{bmatrix} 20 \\ 9 \\ 18 \end{bmatrix}$$

followed by the LDL^H decomposition of **R** using the MATLAB function `[L,D]=ldlt(X)`. This gives

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.5333 & 1 & 0 \\ 0.8667 & -0.5385 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 15 & 0 & 0 \\ 0 & 1.7333 & 0 \\ 0 & 0 & 0.2308 \end{bmatrix}$$

and working through the steps in Table 7.1, we find the LS solution and LSE to be

$$\mathbf{c}_{\text{ls}} = [3.0 \quad -1.5 \quad -1.0]^T \qquad E_{\text{ls}} = 1.5$$

using the following sequence of MATLAB commands

```
k = L\dhat;
cls = L'\k;
Els = sum((y-X'*cls).^2);
```

These results can be verified by using the command `cls=Rhat\dhat`.

7.5.2 LSE FIR Filtering and Prediction

As we stressed in Section 6.3, the fundamental difference between general linear estimation and FIR filtering and prediction, which is the key to the development of efficient order-recursive algorithms, is the shift invariance of the input data vector

$$\mathbf{x}_{m+1}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-m+1)\ x(n-m)]^{\text{T}} \tag{7.5.13}$$

The input data vector can be partitioned as

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \tag{7.5.14}$$

which shows that samples from different times are incorporated as the order is increased. This creates a coupling between order and time updatings that has significant implications in the development of efficient algorithms. Indeed, we can easily see that the matrix

$$\hat{\mathbf{R}}_{m+1} = \sum_{n=N_i}^{N_f} \mathbf{x}_{m+1}(n)\mathbf{x}_{m+1}^{\text{H}}(n) \tag{7.5.15}$$

can be partitioned as

$$\hat{\mathbf{R}}_{m+1} = \begin{bmatrix} \hat{\mathbf{R}}_m & \hat{\mathbf{r}}_m^b \\ \hat{\mathbf{r}}_m^{b\text{H}} & E_m^b \end{bmatrix} = \begin{bmatrix} E_m^f & \hat{\mathbf{r}}_m^{f\text{H}} \\ \hat{\mathbf{r}}_m^f & \hat{\mathbf{R}}_m^f \end{bmatrix} \tag{7.5.16}$$

where

$$\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m + \mathbf{x}_m(N_i-1)\mathbf{x}_m^{\text{H}}(N_i-1) - \mathbf{x}_m(N_f)\mathbf{x}_m^{\text{H}}(N_f) \tag{7.5.17}$$

is the matrix equivalent of (7.2.28). We notice that the relationship between $\hat{\mathbf{R}}_m^f$ and $\hat{\mathbf{R}}_m$, which allows for the development of a complete set of order-recursive algorithms for FIR filtering and prediction, depends on the choice of N_i and N_f, that is, the windowing method selected.

As we discussed in Section 7.3, there are four cases of interest. In the *full-windowing* case ($N_i = 0$, $N_f = N + M - 2$), we have $\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m$ and $\hat{\mathbf{R}}_m$ is Toeplitz. Therefore, all the algorithms and structures developed in Chapter 6 for Toeplitz matrices can be utilized.

In the *prewindowing* case ($N_i = 0$, $N_f = N - 1$), Equation (7.5.17) becomes

$$\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m - \mathbf{x}_m(N-1)\mathbf{x}_m^{\text{H}}(N-1) \tag{7.5.18}$$

Since $\mathbf{x}_m(n) = \mathbf{0}$ for $n \le 0$ (prewindowing), $\hat{\boldsymbol{R}}_m$ is a function of N. If we use the definition

$$\hat{\boldsymbol{R}}_m(N) \triangleq \sum_{n=0}^{N-1} \boldsymbol{x}_m(n)\boldsymbol{x}_m^{\mathrm{H}}(n) \tag{7.5.19}$$

then the time-updating (7.5.18) can be written as

$$\hat{\boldsymbol{R}}_m^{\mathrm{f}} = \hat{\boldsymbol{R}}_m(N-1) = \hat{\boldsymbol{R}}_m(N) - \boldsymbol{x}_m(N-1)\boldsymbol{x}_m^{\mathrm{H}}(N-1) \tag{7.5.20}$$

and the order-updating (7.5.16) as

$$\hat{\boldsymbol{R}}_{m+1}(N) = \begin{bmatrix} \hat{\boldsymbol{R}}_m(N) & \hat{\boldsymbol{r}}_m^{\mathrm{b}}(N) \\ \hat{\boldsymbol{r}}_m^{\mathrm{bH}}(N) & E_m^{\mathrm{b}}(N) \end{bmatrix} = \begin{bmatrix} E_m^{\mathrm{f}}(N) & \hat{\boldsymbol{r}}_m^{\mathrm{fH}}(N) \\ \hat{\boldsymbol{r}}_m^{\mathrm{f}}(N) & \hat{\boldsymbol{R}}_m(N-1) \end{bmatrix} \tag{7.5.21}$$

which has the same form as (6.3.3). Therefore, all order recursions developed in Section 6.3 can be applied in the prewindowing case. However, to get a complete algorithm, we need recursions for the time updatings of the BLP $\boldsymbol{b}_m(N-1) \to \boldsymbol{b}_m(N)$ and $E_m^b(N-1) \to E_m^b(N)$, which can be developed by using the time-recursive algorithms developed in Chapter 9 for LS adaptive filters. The *postwindowing* case can be developed in a similar fashion, but it is of no particular practical interest.

In the *no-windowing* case ($N_i = M-1$, $N_f = N-1$), matrices $\hat{\boldsymbol{R}}_m$ and $\hat{\boldsymbol{R}}_m^{\mathrm{f}}$ depend on both *M* and *N*. Thus, although the development of order recursions can be done as in the prewindowing case, the time updatings are more complicated due to (7.5.17) (Morf et al. 1977). Setting the lower limit to $N_i = M-1$ means that all filters $\mathbf{c}_m$, $1 \le m \le M$, are optimized over the interval $M-1 \le n \le N-1$, which makes the optimum nesting property possible. If we set $N_i = m-1$, each filter $\mathbf{c}_m$ is optimized over the interval $m-1 \le n \le N-1$; that is, it utilizes all the available data. However, in this case, the optimum nesting property $\hat{\boldsymbol{R}}_m = \hat{\boldsymbol{R}}_M^{\lceil m \rceil}$ does not hold, and the resulting order-recursive algorithms are slightly more complicated (Kalouptsidis et al. 1984).

The development of order-recursive algorithms for FBLP least-squares filters and predictors with linear phase constraints, for example, $\boldsymbol{c}_m = \pm \boldsymbol{J}\boldsymbol{c}_m^*$, is more complicated, in general. A review of existing algorithms and more references can be found in Theodoridis and Kalouptsidis (1993).

In conclusion, we notice that order-recursive algorithms are more efficient than the LDL^H decomposition–based solutions only if *N* is much larger than *M*. Furthermore, their numerical properties are inferior to those of the LDL^H decomposition methods; therefore, a bit of extra caution needs to be exercised when order-recursive algorithms are employed.

7.6 Summary

In this chapter we discussed the theory, implementation, and application of linear estimators (combiners, filters, and predictors) that are optimum according to the LSE criterion of performance. The fundamental differences between linear MMSE and LSE estimators are as follows:

- MMSE estimators are designed using ensemble average second-order moments $\boldsymbol{R}$ and $\boldsymbol{d}$; they can be designed prior to operation, and during their normal operation they need only the input signals.
- LSE estimators are designed using time-average estimates $\boldsymbol{R}$ and $\boldsymbol{d}$ of the second-order moments or data matrix $\boldsymbol{X}$ and the desired response vector $\mathbf{y}$. For this reason LSE estimators are sometimes said to be *data-adaptive*. The design and operation of LSE estimators are *coupled* and are usually accomplished by using either of the following approaches:
  - –Collect a block of training data $\boldsymbol{X}_{tr}$ and $\boldsymbol{y}_{tr}$ and use them to design an LSE estimator; use it to process subsequent blocks. Clearly, this approach is meaningful if all blocks have statistically similar characteristics.
  - –For each collected block of data $\boldsymbol{X}$ and $\boldsymbol{y}$, compute the LSE filter $\boldsymbol{c}_{\mathrm{ls}}$ or the LSE estimate $\hat{\boldsymbol{y}}$ (whatever is needed).

There are various numerical algorithms designed to compute LSE estimators and estimates. For well-behaved data and sufficient numerical precision, all these methods produce the same results and therefore provide the same

LSE performance, that is, the same total squared error.

However, when ill-conditioned data, finite precision, or computational complexity is a concern, the choice of the LS computational algorithm is very important.

In conclusion, we emphasize that there are various ways to compute the coefficients of an optimum estimator and the value of the optimum estimate. We stress that the performance of any optimum estimator, as measured by the MMSE or LSE, does *not* depend on the particular implementation as long as we have sufficient numerical precision. Therefore, if we want to investigate how well an optimum estimator performs in a certain application, we can use any implementation, as long as computational complexity is not a consideration.

Problems

7.1 By differentiating (7.2.8) with respect to the vector $\boldsymbol{c}$, show that the LSE estimator $\boldsymbol{c}_{\text{ls}}$ is given by the solution of the normal equations (7.2.12).

7.2 Let the weighted LSE be given by $E_w = \boldsymbol{e}^{\text{H}}\boldsymbol{W}\boldsymbol{e}$, where W is a Hermitian positive definite matrix.

(*a*) By minimizing E_w with respect to the vector $\boldsymbol{c}$, show that the wieghted LSE estimator is given by (7.2.35).

(*b*) Using the LDL^H decomposition $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\text{H}}$, show that the weighted LS criterion corresponds to prefiltering the error or the data.

7.3 Using direct substitution of (7.4.4) into (7.4.5), show that the LS estimator $\boldsymbol{c}_{\text{ls}}^{(i)}$ and the associated LS error $E_{\text{ls}}^{(i)}$ are determined by (7.4.5).

7.4 Consider a linear system described by the difference equation $y(n) = 0.9y(n-1) + 0.1x\ (n-1) + v(n)$, where $x(n)$ is the input signal, $y(n)$ is the output signal, and $v(n)$ is an output disturbance. Suppose that we have collected $N = 1000$ samples of input-output data and that we wish to estimate the system coefficients, using the LS criterion with no windowing. Determine the coefficients of the model $y(n) = ay(n-1) + dx(n-1)$ and their estimated covariance matrix $\hat{\sigma}_e^2\hat{\boldsymbol{R}}^{-1}$ when

(*a*) $x(n) \sim WGN(0,1)$ and $v(n) \sim WGN(0,1)$ and

(*b*) $x(n) \sim WGN(0,1)$ and $v(n) = 0.8v(n-1) + \omega(n)$ is an AR(1) process with $w(n) \sim WGN(0,1)$. Comment upon the quality of the obtained estimates by comparing the matrices $\hat{\sigma}_e^2\hat{\boldsymbol{R}}^{-1}$ obtained in each case.

7.5 Use Lagrange multipliers to show that Equation (7.4.13) provides the minimum of (7.4.8) under the constraint (7.4.9).

7.6 If full windowing is used in LS, then the autocorrelation matrix is Toeplitz. Using this fact, show that in the combined FBLP the predictor is given by

$$\boldsymbol{a}^{\text{fb}} = \frac{1}{2}(\boldsymbol{a} + \boldsymbol{J}\boldsymbol{b}^*)$$

7.7 Consider the noncausal "middle" sample linear signal estimator specified by (7.4.1) with $M = 2L$ and $i = L$.

(*a*) Show that if we apply full windowing to the data matrix, the resulting signal estimator is conjugate symmetric, that is, $\boldsymbol{c}^{(L)} = \boldsymbol{J}\boldsymbol{c}^{(L)*}$. This property does not hold for any other windowing method.

(*b*) Derive the normal equations for the signal estimator that minimizes the total squared error $E^{(L)} = \|\boldsymbol{e}^{(L)}\|^2$ under the constraint $\boldsymbol{c}^{(L)} = \boldsymbol{J}\boldsymbol{c}^{(L)*}$.

(*c*) Show that if we enforce the normal equation matrix to be centro-Hermitian, that is, we use the normal equations

$$(\bar{\boldsymbol{X}}^{\text{H}}\bar{\boldsymbol{X}} + \boldsymbol{J}\bar{\boldsymbol{X}}^{T}\bar{\boldsymbol{X}}^{*}\boldsymbol{J})\boldsymbol{c}^{(L)} = \begin{bmatrix} \boldsymbol{0} \\ E^{(L)} \\ \boldsymbol{0} \end{bmatrix}$$

then the resulting signal smoother is conjugate symmetric.

(*d*) Illustrate parts (*a*) to (*c*), using the data matrix

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

and check which smoother provides the smallest total squared error. Try to justify the obtained answer.

7.8 A useful impulse response for some geophysical signal processing applications is the Mexican hat wavelet

$$g(t) = \frac{2}{\sqrt{3}}\pi^{-1/4}(1-t^2)e^{-t^2/2}$$

which is the second derivative of a Gaussian pulse.

(*a*) Plot the wavelet $g(t)$ and the magnitude and phase of its Fourier transform.

(*b*) By examining the spectrum of the wavelet, determine a reasonable sampling frequency F_s.

(*c*) Design an optimum LS inverse FIR filter for the discrete-time wavelet $g(nT)$, where $T = 1/F_s$. Determine a reasonable value for *M* by plotting the LSE E_M as a function of order *M*. Investigate whether we can improve the inverse filter by introducing some delay n_0. Determine the best value of n_0 and plot the impulse response of the resulting filter and the combined impulse response $g(n) * h(n - n_0)$, which should resemble an impulse.

(*d*) Repeat part (*c*) by increasing the sampling frequency by a factor of 2 and comparing with the results obtained in part (*c*).

7.9 (*a*) Prove Equation (7.5.4) regarding the LDL^{H} decomposition of the augmented matrix $\bar{\boldsymbol{R}}$.

(*b*) Solve the LS estimation problem in Example 7.5.1, using the LDL^{H} decomposition of $\bar{\boldsymbol{R}}$ and the partitionings in (7.5.4).

7.10 Prove the order-recursive algorithm described by the relations given in (7.5.12). Demonstrate the validity of this approach, using the data in Example 7.5.1.

7.11 In this problem, we wish to show that the statistical interpretations of innovation and partial correlation for $w_m(n)$ and k_{m+1} in (7.5.12) hold in a deterministic LSE sense. To this end, suppose that the "partial correlation" between $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{x}}_{m+1}$ is defined using the residual records $\tilde{\boldsymbol{e}}_m = \tilde{\boldsymbol{y}} - \boldsymbol{X}_m \boldsymbol{c}_m$ and $\tilde{\boldsymbol{e}}_m^b = \tilde{\boldsymbol{x}}_{m+1} + \boldsymbol{X}_m \boldsymbol{b}_m$, where $\boldsymbol{b}_m$ is the LSE BLP. Show that $k_{k+1} = \beta_{m+1}/\xi_{m+1}$, where $\beta_{m+1} \triangleq \tilde{\boldsymbol{e}}_m^{\mathrm{H}} \tilde{\boldsymbol{e}}_m^b$ and $\xi_{m+1} = \tilde{\boldsymbol{e}}_m^{b\mathrm{H}} \tilde{\boldsymbol{e}}_m^b$. Demonstrate the validity of these formulas using the data in Example 7.5.1.

7.12 Show that the Cholesky decomposition of a Hermitian positive definite matrix $\boldsymbol{R}$ can be computed by using the following algorithm

$$
\begin{aligned}
&\text{for } j = 1 \text{ to } M \\
&\quad l_{ij} = (r_{ij} - \sum_{k=1}^{j-1} |l_{jk}|^2)^{1/2} \\
&\quad \text{for } i = j+1 \text{ to } M \\
&\quad l_{ij} = (r_{ij} - \sum_{k=1}^{j-1} l_{ik}^* l_{jk}) / l_{jj} \\
&\quad \text{end } i \\
&\text{end } j
\end{aligned}
$$

and write a MATLAB function for its implementation. Test your code using the built-in MATLAB function `chol`.

7.13 In this problem we examine in greater detail the radio-frequency interference cancelation experiment discussed in Section 7.4.3. We first explain the generation of the various signals and then proceed with the design and evaluation of the LS interference canceler.

(*a*) The useful signal is a pointlike target defined by

$$s(t) = \mathrm{d}/\mathrm{d}t \left(1/\mathrm{e}^{-\alpha t/t_r} + \mathrm{e}^{\alpha t/t_f} \right) \triangleq \frac{dg(t)}{dt}$$

where $\alpha = 2.3$, $t_r = 0.4$, and $t_f = 2$. Given that $F_s = 2$ GHz, determine $s(n)$ by computing the samples $g(n) = g(nT)$ in the interval $-2 \le nT \le 6$ ns and then computing the first difference $s(n) = g(n) - g(n-1)$. Plot the signal $s(n)$ and its Fourier transform (magnitude and phase), and check whether the pointlike and wideband assumptions are justified.

(*b*) Generate $N = 4096$ samples of the narrowband interference using the formula

$$z(n) = \sum_{i=1}^{L} A_i \sin(\omega_i n + \varphi_i)$$

and the following information:

```
Fs=2; %All frequencies are measured in GHz.
F=0.1*[0.6 1 1.8 2.1 3 4.8 5.2 5.7 6.1 6.4 6.7 7 7.89.3]';
L=length(F);
om=2*pi*F/Fs;
A=[0.5 1 1 0.5 0.1 0.3 0.5 1 1 1 0.5 0.3 1.5 0.5]';
rand('seed',1954);
phi=2*pi*rand(L,1);
```

(*c*) Compute and plot the the periodogram of $z(n)$ to check the correctness of your code.

(*d*) Generate *N* samples of white Gaussian noise $v(n) \sim WGN(0, 0.1)$ and create the observed signal $x(n) = 5s(n - n_0) + z(n) + v(n)$,

where $n_0 = 1000$. Compute and plot the periodogram of $x(n)$.

(*e*) Design a one-step ahead $(D=1)$ linear predictor with $M=100$ coefficients using the FBLP method with no windowing. Then use the obtained FBLP to clean the corrupted signal $x(n)$ as shown in Figure 7.7. To evaluate the performance of the canceler, generate the plots shown in Figures 7.8 and 7.9.

7.14 Careful inspection of Figure 7.9 indicates that the the *D*-step prediction error filter, that is, the system with input $x(n)$ and output $e^f(n)$, acts as a whitening filter. In this problem, we try to solve Problem 7.13 by designing a practical whitening filter using a power spectral density (PSD) estimate of the corrupted signal $x(n)$.

(*a*) Estimate the PSD $\hat{R}_x^{(PA)}(e^{j\omega_k})$, $\omega_k = 2\pi k/N_{\text{FFT}}$, of the signal $x(n)$, using the method of averaged periodograms. Use a segment length of $L=256$ samples, 50 percent overlap, and $N_{\text{FFT}} = 512$.

(*b*) Since the PSD does not provide any phase information, we shall design a whitening FIR filter with linear phase by

$$\tilde{H}(k) = \frac{1}{\sqrt{\hat{R}_x^{(PA)}(e^{j\omega_k})}} e^{-j\frac{2\pi}{N_{\text{FFT}}}\frac{N_{\text{FFT}}-1}{2}k}$$

where $\tilde{H}(k)$ is the DFT of the impulse response of the filter, that is,

$$\tilde{H}(k) = \sum_{n=0}^{N_{\text{FFT}}-1} h(n) e^{-j\frac{2\pi}{N_{\text{FFT}}}nk}$$

with $0 \le k \le N_{FFT} - 1$.

(*c*) Use the obtained whitening filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 7.8 and 7.9.

(*d*) Repeat part (*c*) with $L=128$, $N_{\text{FFT}} = 512$ and $L=512$, $N_{\text{FFT}} = 1024$ and check whether spectral resolution has any effect upon the performance. *Note:* Information about the design and implementation of FIR filters using the DFT can be found in Proakis and Manolakis (1996).

7.15 Repeat Problem 7.14, using the multitaper method of PSD estimation.

7.16 In this problem we develop an RFI canceler using a symmetric linear smoother with guard samples defined by

$$e(n) = x(n) - \hat{x}(n) \triangleq x(n) + \sum_{k=D}^{M} c_k [x(n-k) + x(n+k)]$$

where $1 \le D < M$ prevents the use of the *D* adjacent samples to the estimation of $x(n)$.

(*a*) Following the approach used in Section 7.4.3, demonstrate whether such a canceler can be used to mitigate RFI and under what conditions.

(*b*) If there is theoretical justification for such a canceler, estimate its coefficients, using the method of LS with no windowing for $M=50$ and $D=1$ for the situation described in Problem 7.13.

(*c*) Use the obtained filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 7.8 and 7.9.

(*d*) Repeat part (c) for $D=2$.

7.17 In Example 5.7.1 we studied the design and performance of an optimum FIR inverse system. In this problem, we design and analyze the performance of a similar FIR LS inverse filter, using training input-output data.

(*a*) First, we generate $N=100$ observations of the input signal $y(n)$ and the noisy output signal $x(n)$. We assume that $x(n) \sim WGN(0,1)$ and $v(n) \sim WGN(0,0.1)$. To avoid transient effects, we generate 200 samples and retain the last 100 samples to generate the required data records.

(*b*) Design an LS inverse filter with $M=10$ for $0 \le D < 10$, using no windowing, and choose the best value of delay *D*.

(*c*) Repeat part (*b*) using full windowing.

(*d*) Compare the LS filters obtained in parts (*b*) and (*c*) with the optimum filter designed in Example 5.7.1. What are your conclusions?

7.18 In this problem we estimate the equalizer discussed in Example 5.8.1, using input-output training data, and we evaluate its performance using Monte Carlo simulation.

(*a*) Generate $N=1000$ samples of input-desired response data $\{x(n), a(n)\}_0^{N-1}$ and use them to estimate the correlation matrix $\hat{\mathbf{R}}_x$ and the cross-correlation vector $\boldsymbol{d}$ between $\mathbf{x}(n)$ and $y(n-D)$. Use $D=7$, $M=11$, and $W=2.9$. Solve the normal equations to determine the LS FIR equalizer and the corresponding LSE.

(*b*) Repeat part (a) 500 times; by changing the seed of the random number generators, compute the average (over the realizations) coefficient vector and average LSE, and compare with the optimum MSE equalizer obtained in Example 5.8.1. What are your conclusions?

(*c*) Repeat parts (a) and (b) by setting $W = 3.1$.

CHAPTER 8

Signal Modeling and Parametric Spectral Estimation

This chapter is a transition from theory to practice. It focuses on the selection of an appropriate model for a given set of data, the estimation of the model parameters, and how well the model actually "fits the data." Although the development of parameter estimation techniques requires a strong theoretical background, the selection of a good model and its subsequent evaluation require the user to have sufficient practical experience and a familiarity with the intended application. We provide complete, detailed algorithms for fitting pole-zero models to data using least-squares techniques. The estimation of all-pole model parameters involves the solution of a linear system of equations, whereas pole-zero modeling requires nonlinear least-squares optimization. The chapter is roughly organized into two separate but related parts.

In the first part, we begin in Section 8.1 by explaining the steps that are required in the model-building process. Then, in Section 8.2, we introduce various least-squares algorithms for the estimation of parameters of direct and lattice all-pole models, provide different interpretations, and discuss some order selection criteria. For pole-zero models we provide, in Section 8.3, a nonlinear optimization algorithm that estimates the parameters of the model by minimizing the least-squares criterion. We conclude this part with Section 8.4 in which we discuss the applications of pole-zero models to spectral estimation and speech processing.

In the second part, we begin with the method of minimum-variance spectral estimation (Capon's method). Then we describe frequency estimation methods based on the harmonic model: the Pisarenko harmonic decomposition and the MUSIC, minimum-norm, and ESPRIT algorithms. These methods are suitable for applications in which the signals of interest can be represented by *complex exponential* or *harmonic models*. Signals consisting of complex exponentials are found in a variety of applications including as formant frequencies in speech processing, moving targets in radar, and spatially propagating signals in array processing.

8.1 The Modeling Process: Theory and Practice

In this section, we discuss the modeling of *real-world* signals using parametric pole-zero (PZ) signal models, whose theoretical properties were discussed in Chapter 3 We focus on PZ (P, Q) models with white input sequences, which are also known as ARMA (P, Q) random signal models. These models are defined by the linear constant-coefficient difference equation

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + \omega(n) + \sum_{k=1}^{Q} d_k \omega(n-k) \tag{8.1.1}$$

where $\omega(n) \sim WN(0, \sigma_\omega^2)$ with $\sigma_\omega^2 < \infty$. The power spectral density (PSD) of the output signal is

$$R(\mathrm{e}^{\mathrm{j}\omega}) = \sigma_\omega^2 \left| \frac{1+\sum_{k=1}^{Q} d_k \,\mathrm{e}^{-\mathrm{j}\omega k}}{1+\sum_{k=1}^{Q} a_k \,\mathrm{e}^{-\mathrm{j}\omega k}} \right|^2 = \sigma_\omega^2 \frac{|D(\mathrm{e}^{-\mathrm{j}\omega})|^2}{|A(\mathrm{e}^{-\mathrm{j}\omega})|^2} \tag{8.1.2}$$

which is a rational function completely specified by the parameters, $\{a_1, a_2, \cdots, a_P\}$, $\{d_1, \cdots, d_Q\}$, and σ_ω^2. We stress that since these models are linear, time-invariant (LTI), the resulting process $x(n)$ is stationary, which is ensured if the corresponding systems are BIBO stable.

The essence of signal modeling and of the resulting parametric spectrum estimation is the following: Given finite-length data $\{x(n)\}_{n=0}^{N-1}$, which can be regarded as a sample sequence of the signal under consideration, we

want to estimate signal model parameters $\{\hat{a}_k\}_1^P$, $\{\hat{b}_k\}_1^Q$, and $\hat{\sigma}_\omega^2$ to satisfy a prescribed criterion. Furthermore, if the parameter estimates are sufficiently accurate, then the following formula

$$\hat{R}(e^{j\omega}) = \hat{\sigma}_\omega^2 \left| \frac{1+\sum_{k=1}^{Q}\hat{d}_k e^{-j\omega k}}{1+\sum_{k=1}^{Q}\hat{a}_k e^{-j\omega k}} \right|^2 = \hat{\sigma}_\omega^2 \frac{|\hat{D}(e^{-j\omega})|^2}{|\hat{A}(e^{-j\omega})|^2} \tag{8.1.3}$$

should provide a reasonable estimate of the signal PSD. A similar argument applies to harmonic signal models and harmonic spectrum estimation in which the model parameters are the amplitudes and frequencies of complex exponentials (see Section 2.1.6).

The development of such models involves the steps shown in Figure 8.1 In this chapter, we assume that we have removed trends, seasonal variations, and other nonstationarities from the data. We further assume that unit poles have been removed from the data by using the differencing approach discussed in Box et al. (1994).

Model selection

In this step, we basically select the structure of the model (direct or lattice), and we make a preliminary decision on the orders P and Q of the model. The most important aid to model selection is the insight and understanding of the signal and the physical mechanism that generates it. Hence, in some applications (e.g., speech processing) physical considerations point to the type and order of the model; when we lack a priori information or we have insufficient knowledge of the mechanism generating the signal, we resort to data analysis methods.

In general, to select a candidate model, we estimate the autocorrelation, partial autocorrelation, and power spectrum from the available data, and we compare them to the corresponding quantities obtained from the theoretical models (see Table 3.1). This preliminary data analysis provides sufficient information to choose a PZ model and some initial estimate for P and Q to start a model building process. Several order selection criteria have been developed that penalize both model misfit and a large number of parameters. Although theoretically interesting and appealing, these criteria are of limited value when we deal with actual signals.

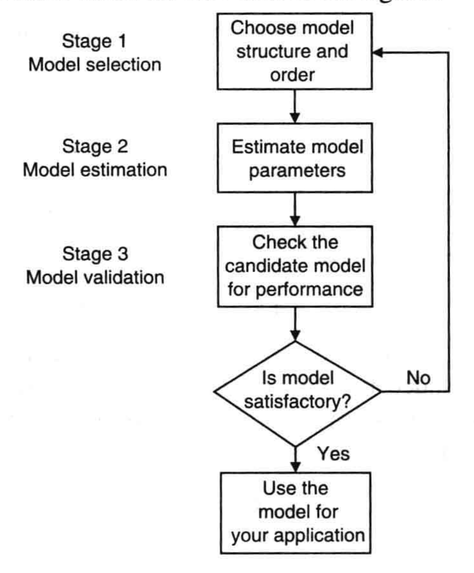


FIGURE 8.1
Steps in the signal model building process.

The model structure influences (1) the complexity of the algorithm that estimates the model parameters and (2) the shape of the criterion function (quadratic or nonquadratic). Therefore, the structure (direct or lattice) is not critical to the performance of the model, and its choice is not as crucial as the choice of the order of the model.

Model estimation

In this step, also known as *model fitting*, we use the available data $\{x(n)\}_0^{N-1}$ to estimate the parameters of the selected model, using optimization of some criterion. Although there are several criteria (e.g., maximum likelihood, spectral matching) that can be used to measure the performance or quality of a PZ model, we concentrate on the

least-squares (LS) error criterion. As we shall see, the estimation of all-pole (AP) models leads to linear optimization problems whereas the estimation of all-zero (AZ) and PZ models requires the solution of nonlinear optimization problems. Parameter estimation for PZ models using other criteria can be found in Kay (1988), Box et al. (1994), Porat (1994), and Ljung (1987).

Model validation

Here we investigate how well the obtained model captures the key features of the data. We then take corrective actions, if necessary, by modifying the order of the model, and repeat the process until we get an acceptable model. The goal of the model validation process is to find out whether the model

- Agrees sufficiently with the observed data
- Describes the "true" signal generation system
- Solves the problem that initiated the design process

Of course, the ultimate test is whether the model satisfies the requirements of the intended application, that is, the objective and subjective criteria that specify the performance of the model, computational complexity, cost, etc. In this discussion, we concentrate on how well the model fits the observed data in an LS error statistical sense.

The existence of any structure in the residual or prediction error signal indicates a misfit between the model and the data. Hence, a key validation technique is to check whether the residual process, which is generated by the inverse of the fitted model, is a realization of white noise. This can be checked by using, among others, the following statistical techniques (Brockwell and Davis 1991; Bendat and Piersol 1986):

Autocorrelation test. It can be shown (Kendall and Stuart 1983) that when N is sufficiently large, the distribution of the estimated autocorrelation coefficients $\hat{\rho}(l) = \hat{r}(l)/\hat{r}(0)$ is approximately Gaussian with zero mean and variance of $1/N$. The approximate 95 percent confidence limits are $\pm 1.96/\sqrt{N}$. Any estimated values of $\hat{\rho}(l)$ that fall outside these limits are "significantly" different from zero with 95 percent confidence. Values well beyond these limits indicate nonwhiteness of the residual signal.

Power spectrum density test. Given a set of data $\{x(n)\}_{n=0}^{N-1}$, the *standardized cumulative periodogram* is defined by

$$\tilde{I}(k) \triangleq \begin{cases} 0 & k < 1 \\ \dfrac{\sum_{i=1}^{k} \hat{R}(e^{j2\pi i/N})}{\sum_{i=1}^{k} \hat{R}(e^{j2\pi i/N})} & 1 \le k \le K \\ 1 & k > K \end{cases} \tag{8.1.4}$$

where K is the integer part of $N/2$. If the process $x(n)$ is white Gaussian noise (WGN), then the random variables $\tilde{I}(k)$, $k = 1, 2, \cdots, K$, are independently and uniformly distributed in the interval $(0,1)$, and the plot of $\tilde{I}(k)$ should be approximately linear with respect to k (Jenkins and Watts 1968). The hypothesis is rejected at level 0.05 if $\tilde{I}(k)$ exits the boundaries specified by

$$\tilde{I}^{(b)}(k) = \frac{k-1}{K-1} \pm 1.36(K-1)^{-1/2} \qquad 1 \le k \le K \tag{8.1.5}$$

Partial autocorrelation test. This test is similar to the autocorrelation test. Given the residual process $x(n)$, it can be shown (Kendall and Stuart 1983) that when N is sufficiently large, the partial autocorrelation sequence (PACS) values $\{k_l\}$ for lag l [defined in (3.2.44)] are approximately independent with distribution $WN(0, 1/N)$. This means that roughly 95 percent of the PACS values fall within the bounds $\pm 1.96/\sqrt{N}$. If we observe values consistently well beyond this range for N sufficiently large, it may indicate nonwhiteness of the signal.

EXAMPLE 9.1.1. To apply the above tests and interpret their results, we consider a WGN sequence $x(n)$. By using the `randn` function, 100 samples of $x(n)$ with zero mean and unit variance were generated. These samples are shown in Figure 8.2. From these samples, the autocorrelation estimates up to lag 40, denoted by $\{\hat{r}(l)\}_{l=0}^{40}$, were computed using the `autoc` function, from which the correlation coefficients $\hat{\rho}(l)$ were obtained. The first 10 coefficients are shown in Figure 8.2 along with the appropriate confidence limits. As expected, the first coefficient at lag 0 is unity while the remaining coefficients are within the

limits.

Next, using the psd function, a periodogram based on 100 samples was computed, from which the cumulative periodogram $\tilde{I}(k)$ was obtained and plotted as a function of the normalized frequency, as shown in Figure 8.2. The confidence limits are also shown. The computed cumulative periodogram is a monotonic increasing function lying within the limits.

Finally, using the durbin function, PACS sequence $\{k_l\}_{l=1}^{40}$ was computed from the estimated correlations and plotted in Figure 8.2. Again all the values for lags $l \geq 1$ are within the confidence limits. Thus all three tests suggest that the 100-point data are almost surely from a white noise sequence.

Although the whiteness of the residuals is a good test for model fitting, it does not provide a definite answer to the problem. Some additional procedures include checking whether

- The criterion of performance decreases (fast enough) as we increase the order of the model.

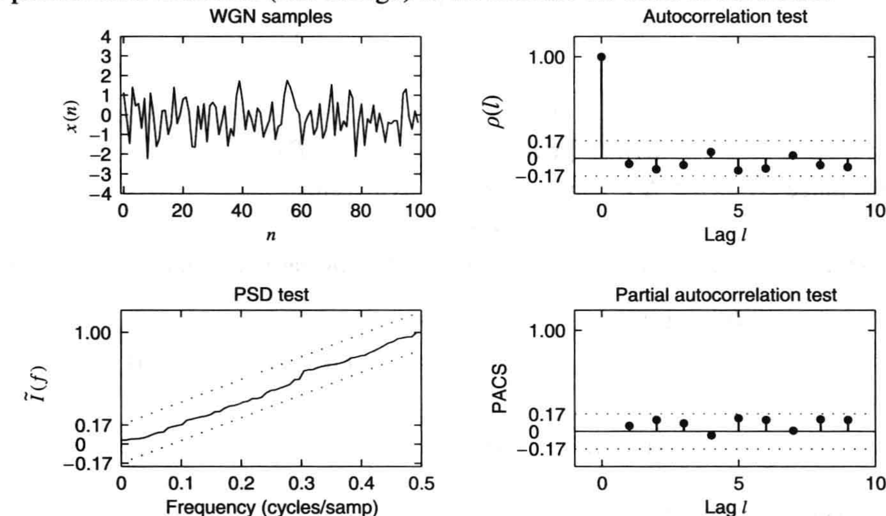


FIGURE 8.2
Validation tests on white Gaussian noise in Example 8.1.1.

- The estimate of the variance of the residual decreases as the number N of observations increases.
- Some estimated parameters that have physical meaning (e.g., reflection coefficients) assume values that make sense.
- The estimated parameters have sufficient accuracy for the intended application.

Finally, to demonstrate that the model is sufficiently accurate for the purpose for which it was designed, we can use a method known as *cross-validation*. Basically, in cross-validation we use one set of data to fit the model and another, statistically independent set of data to test it. Cross-validation is of paramount importance when we build models for control, forecasting, and pattern recognition (Ljung 1987). However, in signal processing applications, such as spectral estimation and signal compression, where the goal is to provide a good fit of the model to the analyzed data, cross-validation is not as useful.

8.2 Estimation of All-Pole Models

We next use the principle of least squares to estimate parameters of all-pole signal models assuming both white and periodic excitations. We also discuss criteria for model order selection, techniques for estimation of all-pole lattice parameters, and the relationship between all-pole estimation methods using the methods of least squares and maximum entropy. The relationship between all-pole model estimation and minimum-variance spectral estimation is explored in Section 8.5.

8.2.1 Direct Structures

Consider the $AR(P_0)$

$$x(n) = -\sum_{k=1}^{P_0} a_k^* x(n-k) + \omega(n) \tag{8.2.1}$$

where $\omega(n) \sim \text{WN}(0, \sigma_w^2)$. The Pth-order linear predictor of $x(n)$ is given by

$$\hat{x}(n) = -\sum_{k=1}^{P} \hat{a}_k^* x(n-k) \tag{8.2.2}$$

and the corresponding prediction error sequence is

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^{P} \hat{a}_k^* x(n-k) \tag{8.2.3}$$

$$= \hat{\boldsymbol{a}}^H \boldsymbol{x}(n) \tag{8.2.4}$$

where $\hat{a}_0 = 1$ and

$$\hat{\boldsymbol{a}} = [1 \ \hat{a}_1 \ \cdots \ \hat{a}_P]^T \tag{8.2.5}$$

$$\boldsymbol{x}(n) = [x(n)x(n-1) \ \cdots \ x(n-P)]^T \tag{8.2.6}$$

Thus the error over the range $N_i \le n \le N_f$ can be expressed as a vector

$$\boldsymbol{e} = \bar{\boldsymbol{X}}\hat{\boldsymbol{a}} \tag{8.2.7}$$

where $\bar{\boldsymbol{X}}$ is the data matrix defined in (7.4.3). For the full-windowing case, the data matrix $\bar{\boldsymbol{X}}$ is given by

$$\bar{\boldsymbol{X}}^H = \begin{bmatrix} x(0) & x(1) & \cdots & x(P) & \cdots & 0 & \cdots & 0 \\ 0 & x(0) & \cdots & x(P-1) & \cdots & x(N-1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x(0) & \cdots & x(N-P) & \cdots & x(N-1) \end{bmatrix} \tag{8.2.8}$$

while for the no-windowing case the data matrix $\bar{\mathbf{X}}$ is

$$\bar{\boldsymbol{X}}^H = \begin{bmatrix} x(P) & x(P+1) & \cdots & x(N-2) & x(N-1) \\ x(P-1) & x(P) & \cdots & x(N-3) & x(N-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x(0) & x(1) & \cdots & x(N-P-2) & x(N-P-1) \end{bmatrix} \tag{8.2.9}$$

Notice that if $P = P_0$ and $\hat{a}_k = a_k$, the prediction error $e(n)$ is identical to the white noise excitation $w(n)$. Furthermore, if $\text{AR}(P_0)$ is minimum-phase, then $w(n)$ is the innovation process of $x(n)$ and $\hat{x}(n)$ is the MMSE prediction of $x(n)$. Thus, we can obtain a good estimate of the model parameters by minimizing some function of the prediction error.

In theory, we minimize the MSE $E\{|e(n)|^2\}$. In practice, since this is not possible, we estimate $\{a_k\}_1^P$ for a given P by minimizing the total squared error

$$E_P = \sum_{n=N_i}^{N_f} |e(n)|^2 = \sum_{n=N_i}^{N_f} \left| x(n) + \sum_{k=1}^{P} \hat{a}_k^* x(n-k) \right|^2 \tag{8.2.10}$$

$$= \sum_{n=N_i}^{N_f} |\hat{\boldsymbol{a}}^H \boldsymbol{x}(n)|^2 = \hat{\boldsymbol{a}}^H \bar{\boldsymbol{X}}^H \bar{\boldsymbol{X}} \hat{\boldsymbol{a}} \tag{8.2.11}$$

over the range $N_i \le n \le N_f$. Hence, we can use the methods discussed in Section 7.4 for the computation of LS linear predictors. In particular, the forward linear predictor coefficient $\{\hat{a}_k\}_{k=1}^P$ and the associated LS error $\hat{\varepsilon}_P$ are obtained by solving the normal equations

$$(\bar{\boldsymbol{X}}^H \bar{\boldsymbol{X}})\hat{\boldsymbol{a}} = \begin{bmatrix} \hat{\varepsilon}_P \\ \mathbf{0} \end{bmatrix} \tag{8.2.12}$$

The solution of (8.2.12) is discussed extensively in Chapter 7.

The least-squares AP(P) parameter estimates have properties similar to those of linear prediction. For example, if the process $\omega(n)$ is Gaussian, the least-squares no-windowing estimates are also maximum-likelihood estimates (Jenkins and Watts 1968). The variance of the excitation process can be obtained from the LS error $\hat{\varepsilon}_P$ by

$$\hat{\sigma}_\omega^2 = \frac{1}{N+P}\hat{\varepsilon}_P = \frac{1}{N+P}\sum_{n=0}^{N+P-1} |e(n)|^2 \qquad \text{full windowing} \tag{8.2.13}$$

or

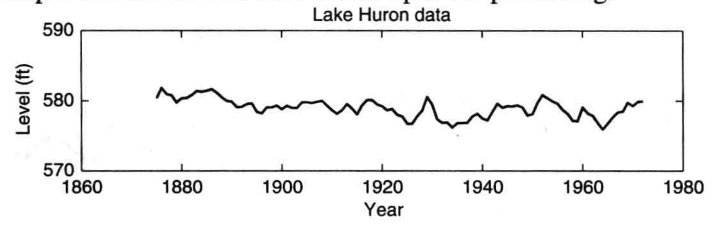
$$\hat{\sigma}_\omega^2 = \frac{1}{N-P}\hat{\varepsilon}_P = \frac{1}{N-P}\sum_{n=P}^{N-1} |e(n)|^2 \qquad \text{no windowing} \tag{8.2.14}$$

for the full-windowing or no-windowing methods, respectively. Furthermore, in the full-windowing case, if the Toeplitz correlation matrix is positive definite, the obtained model is guaranteed to be minimum-phase (see Section 6.4). MATLAB functions

```
[ahat, e, V]=arwin(x, P)and[ahat, e, V]=arls(x, P)
```

are provided that compute the model parameters, the error sequence, and the modeling error using the full-windowing and no-windowing methods, respectively.

We present three examples below to illustrate the all-pole model determination and its use in PSD estimation. The first example uses real data consisting of water-level measurements of Lake Huron from 1875 to 1972. The second example also uses real data containing sunspot numbers for 1770 through 1869. These sunspot numbers have an approximate cycle of period around 10 to 12 years. The Lake Huron and sunspot data are shown in Figure 8.3. The third example generates simulated AR(4) data to estimate model parameters and through them the PSD values. In each case, the mean was computed and removed from the data prior to processing.



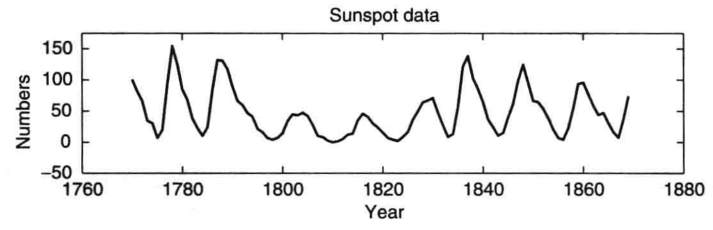


FIGURE 8.3
The Lake Huron and sunspot data used in Examples 8.2.1 and 8.2.2.

EXAMPLE 8.2.1. A careful examination of Lake Huron water-level measurement data indicates that a low-order all-pole model might be a suitable representation of the data. To test this hypothesis, first- and second-order models were considered. Using the full-windowing method, model parameters were computed:

$$\text{First-order} \quad \hat{a}_1 = -0.791, \quad \hat{\sigma}_\omega^2 = 0.5024$$

$$\text{Second-order} \quad \hat{a}_1 = -1.002, \quad \hat{a}_2 = 0.2832, \quad \hat{\sigma}_\omega^2 = 0.4460$$

Using these model parameters, the data were filtered and the residuals were computed. Three tests for checking the whiteness of the residuals as described in Section 8.1 were performed to ascertain the validity of models. In Figure 8.4, we show the residuals, the autocorrelation test, the PSD test, and the partial correlation test for the first-order model. The partial correlation test indicates that the PACS coefficient at lag 1 is outside the confidence limits and thus the first-order model is a poor fit. In Figure 8.5 we show the same plots for the second-order model. Clearly, these tests show that the residuals are approximately white. Therefore, the AR(2) model appears to be a good match to the data.

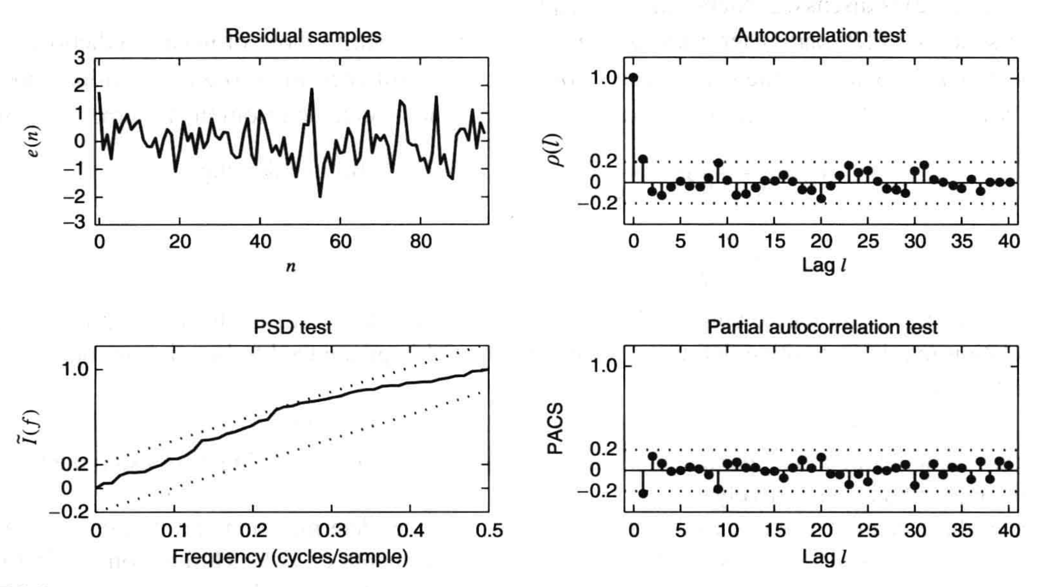


FIGURE 8.4
Validation tests on the first-order model fit to the Lake Huron water-level measurement data in Example 8.2.1.

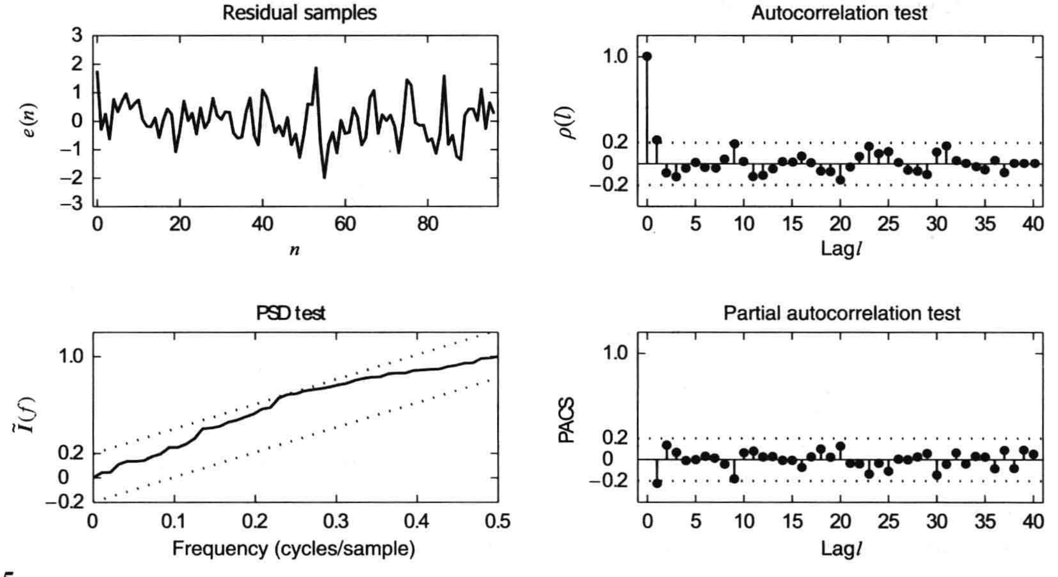


FIGURE 8.5
Validation tests on the second-order model fit to the Lake Huron water-level measurement data in Example 8.2.1.

EXAMPLE 8.2.2. Figure 8.6 shows the PACS coefficients of the sunspot numbers along with the 95 percent confidence limits. Since all PACS values beyond lag 2 fall well inside the limits, a second-order model is a possible candidate for the data. Therefore, the second-order model parameters were estimated from the data to obtain the model

$$x(n) = 1.318x(n-1) - 0.634x(n-2) + \omega(n) \qquad \hat{\sigma}_{\omega}^2 = 289.2$$

In Figure 8.7 we show the residuals obtained by filtering the data along with three tests for its whiteness. The plots show that the estimated model is a reasonable fit to the data. Finally, in Figure 8.8 we show the PSD estimated from the AR(2) model as well as from the periodogram. The periodogram is very noisy and is devoid of any structure. The AR(2) spectrum is smoother and distinctly shows a peak at 0.1 cycle per sampling interval. Since the sampling rate is 1 sampling interval per year, the peak corresponds to 10 years per cycle, which agrees with the observations. Thus the parametric approach to PSD estimation was appropriate.

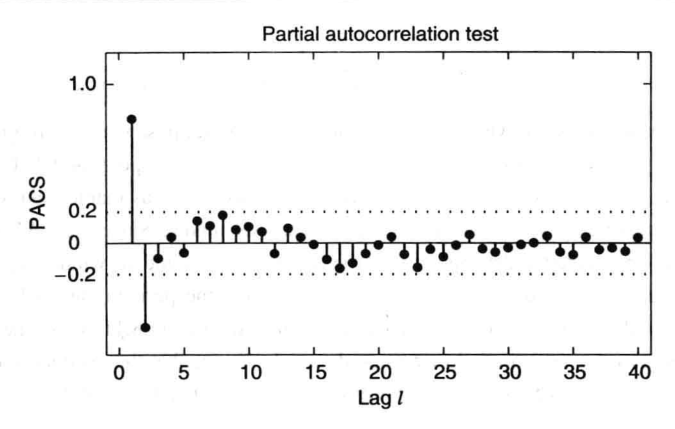


FIGURE 8.6
The PACS values of the sunspot numbers in Example 8.2.2.

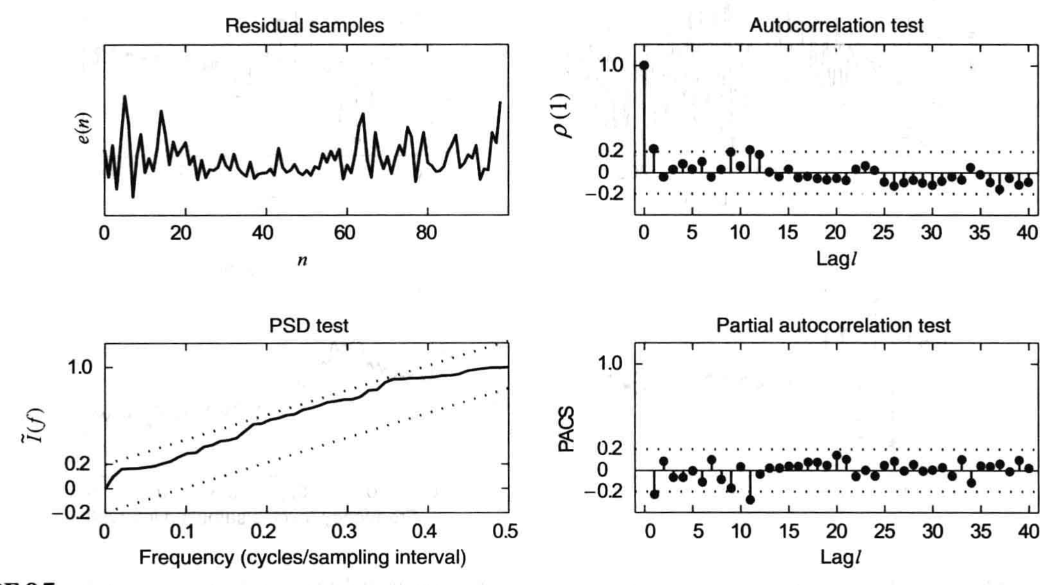


FIGURE 8.7
Validation tests on the second-order model fit to the sunspot numbers in Example 8.2.2.

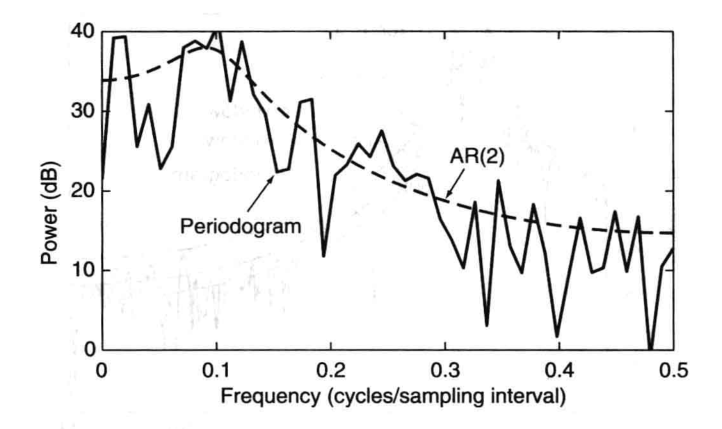


FIGURE 8.8
Comparison of the periodogram and the AR(2) spectrum in Example 8.2.2.

EXAMPLE 8.2.3. We illustrate the least-squares algorithms described above, using the AR(4) process $x(n)$ introduced in Example 4.3.2. The system function of the model is given by

$$H(z)=\frac{1}{1-2.7607z^{-1}+3.8106z^{-2}-2.6535z^{-3}+0.9238z^{-4}}$$

and the excitation is a zero-mean Gaussian white noise with unit variance. Suppose that we are given the $N = 250$ samples of $x(n)$ shown in Figure 8.9 and we wish to model the underlying process by using an all-pole model. To identify a candidate model, we compute the autocorrelation, partial autocorrelation, and periodogram, using the available data. Careful inspection of Figure 8.9 and the signal model characteristics given in Table 3.1 suggests an AR model. Since the PACS plot cuts off around $P = 5$ we choose $P = 4$ and fit an AR(4) model to the data, using both the full-windowing and no-windowing methods. Figure 8.10 shows the actual spectrum of the process, the spectra of the estimated models, and the periodogram. Clearly, the no-windowing estimate provides a better fit because it does not impose any windowing on the data. Figure 8.11 shows the residual, autocorrelation, partial autocorrelation, and periodogram for the no-windowing-based model. We see that the residuals can be assumed uncorrelated with reasonable confidence, which implies that the model captures the second-order statistics of the data.

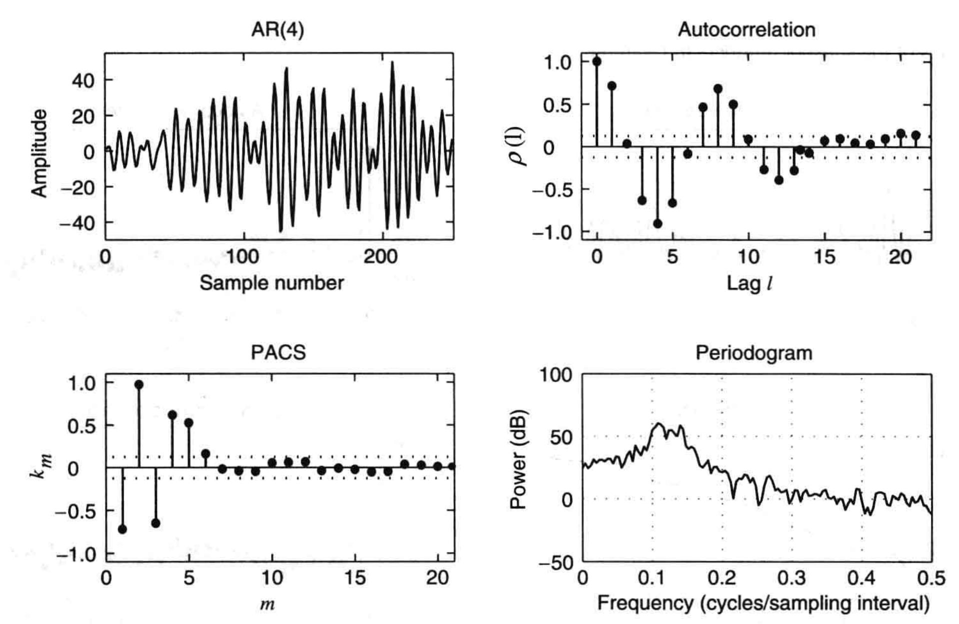


FIGURE 8.9
Data segment from an AR(4) process, and the corresponding autocorrelation, partial autocorrelation, and periodogram.

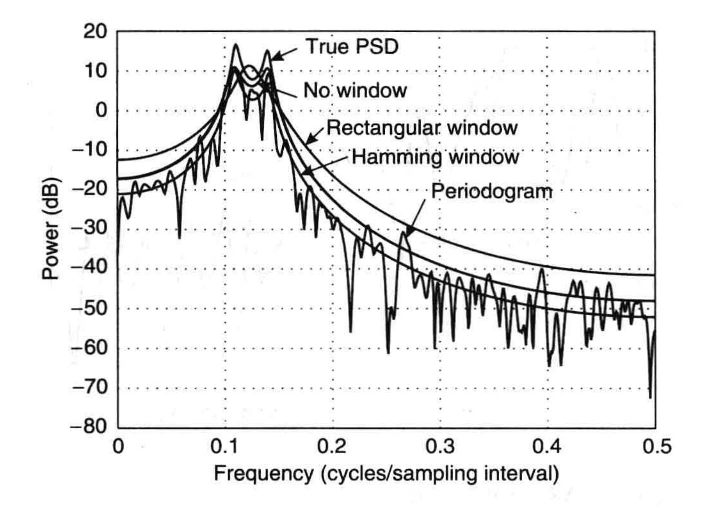


FIGURE 8.10
Periodogram, theoretical AR(4) spectrum, and AR(4) model spectra using full windowing, Hamming windowing, and no windowing.

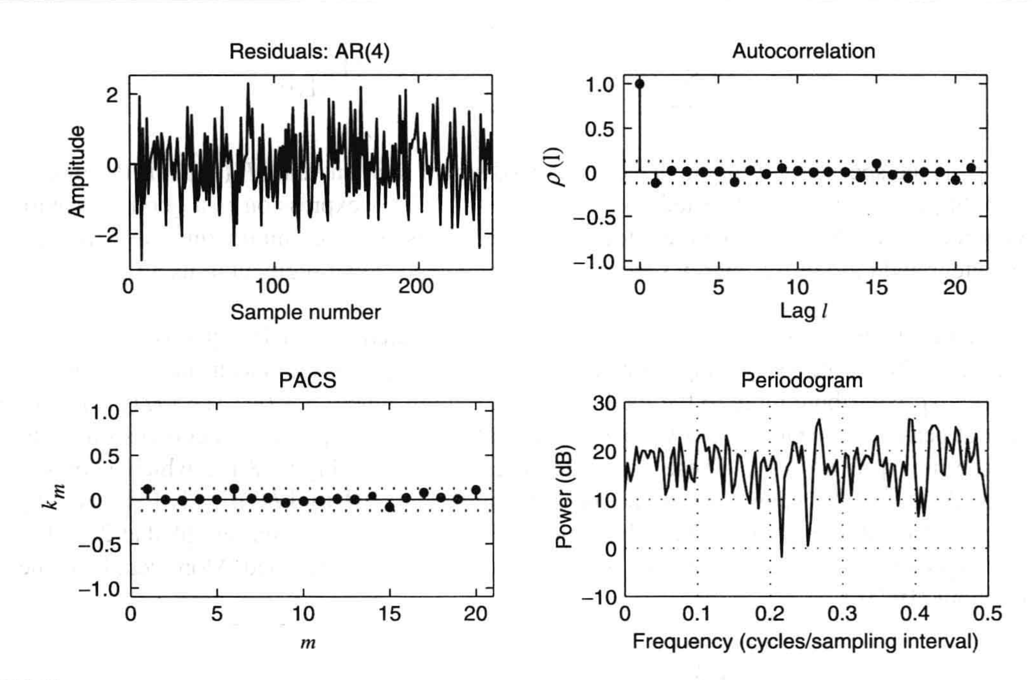


FIGURE 8.11
Residual sequence for the AR(4) data, and the corresponding autocorrelation, partial autocorrelation, and periodogram.

Modified covariance method. The LS method described above to estimate model parameters uses the forward linear predictor and prediction error. There is also another approach that is based on the backward linear predictor. Recall that the backward linear predictor derived from the known correlations is the complex conjugate of the forward predictor (and likewise, the corresponding errors are identical). However, the LS estimators and errors based on the actual data are different because the data read in each direction are *different* from a statistical viewpoint. Hence, it is much more reasonable to consider both forward and backward predictors and to minimize the combined error

$$
\begin{aligned}
\mathcal{E}_P^{\text{fb}} &\triangleq \sum_{n=N_i}^{N_f} [|e^f(n)|^2 + |e^b(n)|^2] \\
&= \sum_{n=N_i}^{N_f} [|\hat{\mathbf{a}}^H \mathbf{x}(n)|^2 + |\hat{\mathbf{a}}^T \mathbf{x}^*(n)|^2] \\
&= \hat{\mathbf{a}}^H \bar{\mathbf{X}}^H \bar{\mathbf{X}} \hat{\mathbf{a}} + \hat{\mathbf{a}}^H \bar{\mathbf{X}}^T \bar{\mathbf{X}}^* \hat{\mathbf{a}}
\end{aligned}
\tag{8.2.15}
$$

subject to the constraint that the first component of $\hat{\mathbf{a}}$ is 1. The minimization of $\mathcal{E}_P^{\text{fb}}$ leads to the set of normal equations

$$
(\bar{\mathbf{X}}^H \bar{\mathbf{X}} + \bar{\mathbf{X}}^T \bar{\mathbf{X}}^*)\hat{\mathbf{a}} = \begin{bmatrix} \hat{\mathcal{E}}_P^{\text{fb}} \\ \mathbf{0} \end{bmatrix}
\tag{8.2.16}
$$

which can be solved efficiently to obtain the model parameters (see Section 7.4.2). This method of using the forward-backward predictors is called the *modified covariance method.* Not only does it have the advantage of minimizing the combined global error, but also since it uses more data in (8.2.16), it gives better estimates and lower error. A similar minimization approach, but implemented at each local stage, is used in Burg's method, which is discussed in Section 8.2.2.

Frequency-domain interpretation. In the case of full windowing, by using Parseval's theorem, the error energy can be written as

$$\varepsilon = \sum_{n=-\infty}^{\infty} |e(n)|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|X(e^{j\omega})|^2}{|\hat{H}(e^{j\omega})|^2} d\omega \tag{8.2.17}$$

where $|X(e^{j\omega})|^2$ is the spectrum of the modeled windowed signal segment and $\hat{H}(e^{j\omega})$ is the frequency response of the estimated all-pole model [or estimated spectrum of $x(n)$]. This expression is a good approximation for the other windowing methods if $N \gg P$. Since the integrand in (8.2.17) is positive, minimizing the error ε is equivalent to minimizing the integrated ratio of the energy spectrum of the modeled signal segment to its all-pole-based spectrum.

The presence of this ratio in (8.2.17) has three additional consequences. (1) The quality of the spectral matching is *uniform* over the whole frequency range, irrespective of the shape of the spectrum. (2) Since regions where $|X(e^{j\omega})| > |\hat{H}(e^{j\omega})|$ contribute more to the total error than regions where $|X(e^{j\omega})| < |\hat{H}(e^{j\omega})|$ do, the match is better near spectral peaks than near spectral valleys. (3) The all-pole model provides a good estimate of the *envelope* of the signal spectrum $|X(e^{j\omega})|^2$. These properties are apparent in Figure 8.12, which shows a comparison between $20\log|X(e^{j\omega})|$ (obtained using the periodogram) and $20\log|\hat{H}(e^{j\omega})|$ [obtained by an AP(28) model fitted using full windowing] for a 20-ms, Hamming windowed, speech signal sampled at 20 kHz. Note that the slope of $|\hat{H}(e^{j\omega})|$ is always zero at frequencies $\omega = 0$ and $\omega = \pi$, as expected. More details on these issues can be found in Makhoul (1975b).

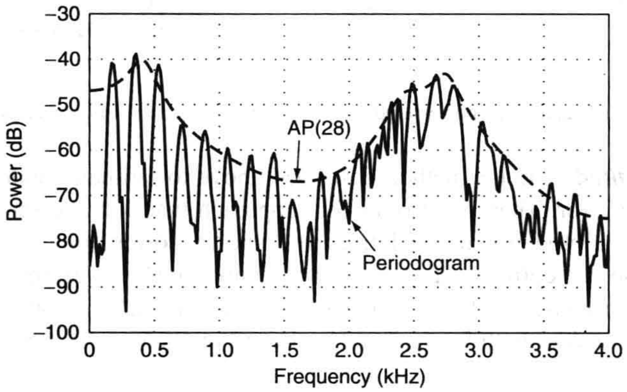


FIGURE 8.12
Illustration of the spectral envelope matching property of all-pole models.

The error energy (8.2.17) is also related to the Itakura-Saito (IS) distortion measure, which is given by

$$d_{IS}(R_1, R_2) \triangleq \frac{1}{2\pi}\int_{-\pi}^{\pi} [\exp V(e^{j\omega}) - V(e^{j\omega}) - 1] d\omega \tag{8.2.18}$$

where $R_1(e^{j\omega})$ and $R_2(e^{j\omega})$ are two spectra, and

$$V(e^{j\omega}) \triangleq \log R_1(e^{j\omega}) - \log R_2(e^{j\omega}) \tag{8.2.19}$$

Indeed, we can show that

$$d_{IS}(R_1, R_2) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{R_1(e^{j\omega})}{R_2(e^{j\omega})} d\omega - \log\frac{\sigma_1^2}{\sigma_2^2} - 1 \tag{8.2.20}$$

where σ_1^2 and σ_2^2 are the variances of the innovation sequences corresponding to the factorization of spectra $R_1(e^{j\omega})$ and $R_2(e^{j\omega})$, respectively. More details can be found in Rabiner and Juang (1993).

Order selection criteria. The order of an all-pole signal model plays an important role in the modeling problem. It determines the number of parameters to be estimated and hence the computational complexity of the algorithm. But more importantly, it affects the quality of the spectrum estimates. If a much lower order is selected, then the resulting spectrum will be smooth and will display poor resolution. If a much larger order is used, then the spectrum may contain spurious peaks at best and a phenomenon called *spectrum splitting* at worst, in which a single peak is split into two separate and distinct peaks (Hayes 1996).

Several criteria have been proposed over the years for model order selection; however, in practice nothing surpasses the graphical approach outlined in Examples 8.2.1 and 8.2.2 combined with the experience of the user. Therefore, we only provide a brief summary of some well-known criteria and refer the interested reader to Kay (1988), Porat (1994), and Ljung (1987) for more details. The simplest approach would be to monitor the modeling error and then select the order at which this error enters a steady state. However, for all-pole models, the modeling error is monotonically decreasing, which makes this approach all but impossible. The general idea behind the suggested criterion is to introduce a penalty function in the modeling error that increases with the model order P. We present the following four criteria that are based on the above general idea.

FPE criterion. The *final prediction error (FPE)* criterion, proposed by Akaike (1970), is based on the function

$$FPE(P) = \frac{N+P}{N-P}\hat{\sigma}_P^2 \tag{8.2.21}$$

where $\hat{\sigma}_P^2$ is the modeling error [or variance of the residual of the estimated AP(P) model]. We note that the term $\hat{\sigma}_P^2$ decreases or remains the same with increasing P, whereas the term $(N+P)/(N-P)$ accounts for the increase in $\hat{\sigma}_P^2$ due to inaccuracies in the estimated parameters and increases with P. Clearly, FPE(P) is an inflated version of $\hat{\sigma}_P^2$. The FPE order selection criterion is to choose P that will minimize the function in (8.2.21).

AIC. The *Akaike information criterion (AIC)*, also introduced by Akaike (1974), is based on the function

$$AIC(P) = N \log \hat{\sigma}_P^2 + 2P \tag{8.2.22}$$

It is a very general criterion that provides an estimate of the Kullback-Leibler distance (Kullback 1959) between an assumed and the true probability density function of the data. The performances of the FPE criterion and the AIC are quite similar.

MDL criterion. The *minimum description length (MDL)* criterion was proposed by Risannen (1978) and uses the function

$$MDL(P) = N \log \hat{\sigma}_P^2 + P \log N \tag{8.2.23}$$

The first term in (8.2.23) decreases with P, but the second penalty term increases. It has been shown (Risannen 1978) that this criterion provides a consistent order estimate in that as the probability that the estimated order is equal to the true order approaches 1, the data length N tends to infinity.

CAT. This criterion is based on Parzen's *criterion autoregressive transfer (CAT)* function (Parzen 1977), which is given by

$$CAT(P) = \frac{1}{N}\sum_{k=1}^{P}\frac{N-k}{N\hat{\sigma}_k^2} - \frac{N-P}{N\hat{\sigma}_P^2} \tag{8.2.24}$$

This criterion is asymptotically equivalent to the AIC and the MDL criteria.

Basically, all order selection criteria add to the variance of the residuals a term that grows with the order of the model and estimate the order of the model by minimizing the resulting criterion. However, when $P \ll N$ which is the case in many practical applications, the criterion does not exhibit a clear minimum that makes the order selection process difficult (see Problem 8.1).

8.2.2 Lattice Structures

We noted in Section 6.5 that a prediction error filter, and hence the AP model, can also be implemented by using a lattice structure. The Pth-order forward prediction error $e(n) = e_P^f(n)$ and the total squared error

$$\varepsilon_P = \sum_{n=N_i}^{N_f} |e(n)|^2 \tag{8.2.25}$$

are nonlinear functions of the lattice parameters k_m, $0 \le m \le P-1$. For example, if $P=2$, we have

$$e_2^f(n) = x(n) + (k_0^* + k_0 k_1^*)x(n-1) + k_1^* x(n-2)$$

which shows that $e_2^f(n)$ depends on the product $k_0 k_1^*$. Thus, fitting an all-pole lattice model by minimizing ε_P

with respect to k_m, $0 \le m \le P-1$, leads to a difficult nonlinear optimization problem.

We can avoid this problem by replacing the above "global" optimization with P "local" optimizations from $m=1$ to P, one for each stage of the lattice. From the lattice equations

$$e_m^f(n) = e_{m-1}^f(n) + k_{m-1}^* e_{m-1}^b(n-1) \tag{8.2.26}$$

$$e_m^b(n) = e_{m-1}^b(n-1) + k_{m-1} e_{m-1}^f(n) \tag{8.2.27}$$

we see that the mth-order prediction errors depend on the coefficient k_{m-1} only. Furthermore, the values of $e_{m-1}^f(n)$ and $e_{m-1}^b(n)$ have been computed by using k_{m-2}, which has been determined from the optimization step at the previous stage.

Hence, to minimize the forward prediction error

$$\varepsilon_m^{\text{f}} = \sum_{n=N_i}^{N_f} |e_m^f(n)|^2 \tag{8.2.28}$$

we substitute (8.2.26) into (8.2.28) and differentiate with respect to k_{m-1}^*. This leads to the following optimum value of k_{m-1}

$$k_{m-1}^{\text{FP}} = -\frac{\beta_{m-1}^{\text{fb}}}{\varepsilon_{m-1}^{\text{b}}} \tag{8.2.29}$$

where

$$\beta_m^{\text{fb}} = \sum_{n=N_{\text{i}}}^{N_{\text{f}}} [e_m^{\text{f}}(n)]^* e_m^{\text{b}}(n-1) \tag{8.2.30}$$

and

$$\varepsilon_m^{\text{b}} = \sum_{n=N_{\text{i}}}^{N_{\text{f}}} |e_m^{\text{b}}(n-1)|^2 \tag{8.2.31}$$

Similarly, minimization of the backward prediction error (8.2.31) gives

$$k_{m-1}^{\text{BP}} = -\frac{\beta_{m-1}^{\text{fb}}}{\varepsilon_{m-1}^{\text{f}}} \tag{8.2.32}$$

Burg (1967) suggested the estimation of k_{m-1} by minimizing

$$\varepsilon_m^{\text{fb}} = \sum_{n=N_{\text{i}}}^{N_{\text{f}}} \{|e_m^{\text{f}}(n)|^2 + |e_m^{\text{b}}(n)|^2\} \tag{8.2.33}$$

at each stage of the lattice.[1] Indeed, substituting (8.2.26) and (8.2.27) in the last equation, we obtain the relationship

$$\varepsilon_m^{\text{fb}} = (1+|k_{m-1}|^2)\varepsilon_{m-1}^{\text{f}} + 4\,\text{Re}(k_{m-1}^*\beta_{m-1}^{\text{fb}}) + (1+|k_{m-1}|^2)\varepsilon_{m-1}^{\text{b}} \tag{8.2.34}$$

If we set $\partial\varepsilon_m^{\text{fb}}/\partial k_{m-1}^* = 0$, we obtain the following estimate of k_{m-1}:

$$k_{m-1}^{\text{B}} = -\frac{\beta_{m-1}^{\text{fb}}}{\frac{1}{2}(\varepsilon_{m-1}^{\text{f}} + \varepsilon_{m-1}^{\text{b}})} = \frac{2k_{m-1}^{\text{FM}}k_{m-1}^{\text{BM}}}{k_{m-1}^{\text{FM}} + k_{m-1}^{\text{BM}}} \tag{8.2.35}$$

We note that k_{m-1}^{B} is the harmonic mean of k_{m-1}^{FP} and k_{m-1}^{BP}. We also stress that the obtained model is *different* from the one resulting from the forward-backward least-squares (FBLS) method through global optimization [see (8.2.11)].

Itakura and Saito (1971) proposed an estimate of k_{m-1} based on replacing the theoretical ensemble averages in

---

[1]This approach should not be confused with the maximum entropy method introduced also by Burg and discussed later.

(6.5.24) by time averages. Their estimate is given by

$$k_{m-1}^{\mathrm{IS}} = -\frac{\beta_{m-1}^{\mathrm{fb}}}{\sqrt{\varepsilon_{m-1}^{\mathrm{f}}\varepsilon_{m-1}^{\mathrm{b}}}} = \mathrm{sign}(k_{m-1}^{\mathrm{FP}} \text{or } k_{m-1}^{\mathrm{BP}})\sqrt{k_{m-1}^{\mathrm{FP}}k_{m-1}^{\mathrm{BP}}} \tag{8.2.36}$$

and is also known as the geometric mean method. Since it can be shown that

$$|k_{m-1}^{\mathrm{B}}| \le |k_{m-1}^{\mathrm{IS}}| \le 1 \tag{8.2.37}$$

both estimates result in minimum-phase models (see Problem 8.2). From (8.2.36) and (8.2.37) we conclude that if $|k_{m-1}^{\mathrm{FP}}| < 1$, then $|k_{m-1}^{\mathrm{BP}}| > 1$ and vice versa; that is, if the FLP is minimum-phase, then the BLP is maximum-phase and vice versa. Several other estimates are discussed in Makhoul (1977) and Viswanathan and Makhoul (1975).

In all previous methods, we use *no windowing*; that is, we set $N_i = m$ and $N_f = N-1$. If we use data windowing, all the above estimates are identical to the data windowing estimates obtained using the algorithm of Levinson-Durbin (see Problem 8.3).

The variance of the residuals can be estimated by

$$\hat{\sigma}_m^2 = \frac{1}{2}\frac{E_m^{\mathrm{fb}}}{N-m} \tag{8.2.38}$$

which for large values of N (see Problem 8.12) can be approximated by

$$\hat{\sigma}_m^2 = \hat{\sigma}_{m-1}^2(1-|k_{m-1}|^2) \tag{8.2.39}$$

where

$$\hat{\sigma}_0^2 = \frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2 \tag{8.2.40}$$

The computations for the lattice estimation methods are summarized in Table 8.1, and the algorithms are implemented by the function `[k,var] = aplatest(x, P)`.

TABLE 8.1
Algorithm for estimation of AP lattice parameters.

1. **Input:** $x(n)$ for $N_i \le n \le N_f$
2. **Initialization**
   a. $e_0^{\mathrm{f}}(n) = e_0^{\mathrm{b}}(n) = x(n)$.
   b. Compute β_0^{fb}, E_0^{f}, and E_0^{b} from $x(n)$
   c. Compute k_1^{FP} and k_1^{BP}
   d. Compute either k_1^{IS} or k_1^{B} from k_1^{FP} and k_1^{BP}
   e. Apply the first stage of the lattice to $x(n)$ using either k_1^{IS} or k_1^{B} to obtain $e_1^{\mathrm{f}}(n)$ and $e_1^{\mathrm{b}}(n)$.
3. **For** $m = 2, 3, \cdots, M$
   a. Compute $\beta_{m-1}^{\mathrm{fb}}$, E_{m-1}^{f}, and E_{m-1}^{b} from $e_{m-1}^{\mathrm{f}}(n)$ and $e_{m-1}^{\mathrm{b}}(n)$
   b. Compute k_{m-1}^{FP} and k_{m-1}^{BP}.
   c. Compute either k_{m-1}^{FP} or k_{m-1}^{B} from k_{m-1}^{FP} and k_{m-1}^{BP}
   d. Apply the m th stage of the lattice to $e_{m-1}^{\mathrm{f}}(n)$ and $e_{m-1}^{\mathrm{b}}(n)$ using either k_{m-1}^{IS} or k_{m-1}^{B} to obtain $e_m^{\mathrm{f}}(n)$ and $e_m^{\mathrm{b}}(n)$.
4. **Output:** Either k_m^{IS} or k_m^{B} for $m = 1, 2, \cdots, M$ and $e_m^{\mathrm{f}}(n)$ and $e_m^{\mathrm{b}}(n)$.

8.2.3 Maximum Entropy Method

We next show how LS all-pole modeling is related to Burg's method of maximum entropy. To this end, suppose that $x(n)$ is a normal, stationary process with zero mean. The M-dimensional complex-valued vector $\boldsymbol{x} \triangleq \boldsymbol{x}_M(n)$ obeys a normal distribution

$$p(\boldsymbol{x}) = \frac{1}{\pi^M \det \boldsymbol{R}}\exp(-\boldsymbol{x}^{\mathrm{H}}\boldsymbol{R}^{-1}\boldsymbol{x}) \tag{8.2.41}$$

where $\boldsymbol{R}$ is a Toeplitz correlation matrix. By definition, its entropy is given by

$$\mathscr{H}(\boldsymbol{x}) \triangleq -E\{\log p(\boldsymbol{x})\} = M \log \pi + \log(\det \boldsymbol{R}) + M \tag{8.2.42}$$

because $E\{\boldsymbol{x}^H \boldsymbol{R}^{-1} \boldsymbol{x}\} = M$. If the process $x(n)$ is regular, that is, $|k_m| < 1$ for all m, we have

$$\det \boldsymbol{R} = \prod_{m=0}^{M-1} P_m \qquad \text{and} \qquad P_m = r(0) \prod_{j=1}^{m} (1 - |k_j|^2) \tag{8.2.43}$$

where $P_m = P_m^{\mathrm{f}} = P_m^{\mathrm{b}}$ (see Section 6.4). If we substitute (8.2.43) into (8.2.42), we obtain

$$\mathscr{H}(\boldsymbol{x}) = M \log \pi + M + M \log r(0) + \sum_{m=1}^{M-1} (M - m) \log(1 - |k_m|^2) \tag{8.2.44}$$

which expresses the entropy in terms of $r(0)$ and the PACS k_m, $1 \le m \le M \le \infty$ [recall that any parametric model can be specified by $r(0)$ and the PACS]. Suppose now that we are given the first P+1 values $r(0)$, $r(1)$, $\cdots$, $r(P)$ of the autocorrelation sequence and we wish to find a model, by choosing the remaining values $r(l), l > P$, so that the entropy is maximized. From (8.2.44), we see that the entropy is maximized if we choose $k_m = 0$ for $m > P$, that is, by modeling the process $x(n)$ by an AR(P) model. In conclusion, among all regular Gaussian processes with the same first $P+1$ autocorrelation values, the AR(P) process has the maximum entropy. Any other choices for k_m, $m > P$, that satisfy the condition $|k_m| < 1$ lead to a valid extension of the autocorrelation sequence. The "extended" values $r(l)$, $l > P$, can be obtained by using the inverse Levinson-Durbin or the inverse Schür algorithm (see Chapter 6). The relation between autoregressive modeling and the principle of maximum entropy, known as the *maximum entropy method*, was introduced by Burg (1967, 1975). We note that the above proof, given in Porat (1994), is different from the original proof provided by Burg (Burg 1975; Therrien 1992). An interesting discussion of various arguments in favor of and against the maximum entropy method can be found in Makhoul (1986).

8.2.4 Excitations with Line Spectra

When the excitation of a parametric model has a spectrum with lines at L frequencies ω_m, the spectrum of the output signal provides information about the frequency response of the model at these frequencies only. For simplicity, assume equidistant samples at frequencies $\omega_m = 2\pi m/L$, $0 \le m \le L-1$. Given a set of values $R_x(\mathrm{e}^{\mathrm{j}\omega_m}) = |X(\mathrm{e}^{\mathrm{j}\omega_m})|^2$, we wish to find an AP($P$) model whose spectrum $\hat{R}_h(\mathrm{e}^{\mathrm{j}\omega})$ matches $R_x(\omega_m)$ at the given frequencies, by minimizing the criterion

$$\tilde{\varepsilon} = \frac{d_0}{L} \sum_{m=1}^{L} \frac{R_x(\mathrm{e}^{\mathrm{j}\omega_m})}{\hat{R}_h(\mathrm{e}^{\mathrm{j}\omega_m})} \tag{8.2.45}$$

which is the discrete version of (8.2.17) and d_0 is the gain of the model (see Section 3.2). The minimization of (8.2.45) with respect to the model parameters $\{a_k\}$ results in the Yule-Walker equations

$$\sum_{k=0}^{P} a_k^* \tilde{r}(i-k) = \begin{cases} \tilde{\varepsilon} & i = 0 \\ 0 & 1 \le i \le P \end{cases} \tag{8.2.46}$$

where

$$\tilde{r}(l) = \frac{1}{L} \sum_{m=1}^{L} R_x(\mathrm{e}^{\mathrm{j}\omega_m}) \mathrm{e}^{\mathrm{j}\omega_m} \tag{8.2.47}$$

For continuous spectra, linear prediction uses the autocorrelation

$$r(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(\mathrm{e}^{\mathrm{j}\omega}) \mathrm{e}^{\mathrm{j}\omega} \, \mathrm{d}\omega \tag{8.2.48}$$

which is related to $\tilde{r}(l)$ by

$$\tilde{r}(l) = \sum_{m=-\infty}^{\infty} r(l - Lm) \tag{8.2.49}$$

that is, $\tilde{r}(l)$ is an aliased version of $r(l)$. We have seen that linear prediction equates the autocorrelation of the AP(P) model to the autocorrelation of the modeled signal for the first P+1 lags. Hence, when we use linear prediction for a signal with line spectra, the autocorrelation of the all-pole model will be matched to $\tilde{r}(l) \neq r(l)$ and will always result in a model different from the original. Clearly, the correlation matching condition cannot compensate for the autocorrelation aliasing, which becomes more pronounced as L decreases. This phenomenon, which is severe for voiced sounds with high pitch, is illustrated in Problem 8.13. A method that provides better estimates, by minimizing a discrete version of the Itakura-Saito error measure, has been developed for both AP and PZ models by El-Jaroudi and Makhoul (1991, 1989).

8.3 Estimation of Pole-zero Models

The estimation of $\mathrm{PZ}(P,Q)$ model parameters for $Q \neq 0$ leads to a nonlinear LS optimization problem. As a result, a vast number of suboptimum methods, with reduced computational complexity, have been developed to avoid this problem. For example, some techniques estimate the AP(P) and AZ(Q) parameters separately. However, today the availability of high-speed computers has made exact least-squares the method of choice. Since the nonlinear LS optimization with respect to complex vectors and its conjugate is inherently difficult, and since this optimization does not provide any additional insight into the solution technique, we assume, in this section, that the quantities are real-valued. Furthermore, most of the real-world applications of pole-zero models almost always involve real-valued signals and systems. The extension to the complex-valued case is straightforward.

Consider the $\mathrm{PZ}(P,Q)$ model

$$x(n) = -\sum_{k=1}^{P} a_k x(n-k) + \omega(n) + \sum_{k=1}^{Q} d_k \omega(n-k) \tag{8.3.1}$$

where $\omega(n) \sim \mathrm{WN}(0,\sigma_\omega^2)$. Using vector notation, we can express (8.3.1) as

$$x(n) = \mathbf{z}^{\mathrm{T}}(n-1)\mathbf{c}_{\mathrm{pz}} + \omega(n) \tag{8.3.2}$$

where
$$z(n) \triangleq [-x(n) \ \cdots \ -x(n-P+1) \ w(n) \ \cdots \ w(n-Q+1)]^{\mathrm{T}} \tag{8.3.3}$$

and
$$\boldsymbol{c}_{\mathrm{pz}} = [\boldsymbol{a}^{\mathrm{T}} \ \boldsymbol{d}^{\mathrm{T}}] = [a_1 \ \cdots \ a_P d_1 \ \cdots \ d_Q]^{\mathrm{T}} \tag{8.3.4}$$

8.3.1 Known Excitation

Assume for a moment that the excitation $w(n)$ is known. Then we can predict $x(n)$ from past values, using the following linear predictor

$$\hat{x}(n) = \boldsymbol{z}^{\mathrm{T}}(n-1)\boldsymbol{c} \tag{8.3.5}$$

where
$$\mathbf{c} = [\hat{a}_1 \ \cdots \ \hat{a}_P \hat{d}_1 \ \cdots \ \hat{d}_Q]^{\mathrm{T}} \tag{8.3.6}$$

are the predictor parameters. The prediction error

$$e(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{z}^{\mathrm{T}}(n-1)\mathbf{c} \tag{8.3.7}$$

equals $w(n)$ *if* $\mathbf{c}=\mathbf{c}_{\mathrm{pz}}$. Minimization of the total squared error

$$\varepsilon(\boldsymbol{c}) \triangleq \sum_{n=N_{\mathrm{i}}}^{N_{\mathrm{f}}} e^2(n) \tag{8.3.8}$$

leads to the following linear system of equations

$$\hat{\boldsymbol{R}}_z \boldsymbol{c} = \hat{\boldsymbol{r}}_z \tag{8.3.9}$$

where
$$\hat{R}_z = \sum_{n=N_i}^{N_f} z(n-1)z^T(n-1) \tag{8.3.10}$$

and
$$\hat{r}_z = \sum_{n=N_i}^{N_f} z(n-1)x(n) \tag{8.3.11}$$

Usually, we use residual windowing, which implies that $N_i = \max(P,Q)$ and $N_f = N-1$. Since the matrix $\hat{\mathbf{R}}_z$ is symmetric and positive semidefinite, we can solve (8.3.9) using LDL^H decomposition. Thus, if we know the excitation $\omega(n)$, the least-squares estimation of the PZ(P,Q) model parameters reduces to the solution of a linear system of equations. An estimate of the input variance is given by

$$\hat{\sigma}_\omega^2 = \frac{1}{N - \max(P,Q)} \sum_{n=\max(P,Q)}^{N-1} e^2(n) \tag{8.3.12}$$

This method, which is implemented by the function `pzls.m`, is known as the *equation-error method* and can be used to identify a system from input-output data (Ljung 1987) (see Problem 8.14).

8.3.2 Unknown Excitation

In most applications, the excitation $w(n)$ is never known. However, we can obtain a good estimate of $x(n)$ by replacing $w(n)$ by $e(n)$ in (8.3.5). This makes a natural choice if the model used to obtain $e(n)$ is reasonably accurate. The prediction error is then given by

$$e(n) = x(n) - \hat{x}(n) = x(n) - \hat{z}^T(n-1)\mathbf{c} \tag{8.3.13}$$

where
$$\hat{\mathbf{z}}(n) \triangleq [-x(n) \ \cdots \ -x(n-P+1) \ e(n) \ \cdots \ e(n-Q+1)]^T \tag{8.3.14}$$

If we write (8.3.13) explicitly

$$e(n) = -\sum_{k=1}^{Q} \hat{d}_k e(n-k) + x(n) + \sum_{k=1}^{P} \hat{a}_k x(n-k) \tag{8.3.15}$$

we see that the prediction error is obtained by exciting the inverse model with the signal $x(n)$. Hence, the inverse model has to be stable. To satisfy this condition, we require the estimated model to be minimum-phase.

The recursive computation of $e(n)$ by (8.3.15) makes the prediction error a nonlinear function of the model parameters. To illustrate this, consider the prediction error for a first-order model, that is, for $P = Q = 1$

$$e(n) = x(n) + \hat{a}_1 x(n-1) - \hat{d}_1 e(n-1)$$

Assuming $e(0) = 0$, we have for $n = 1, 2, 3$

$$\begin{aligned}
e(1) &= x(1) + \hat{a}_1 x(0) \\
e(2) &= x(2) + \hat{a}_1 x(1) - \hat{d}_1 e(1) \\
&= x(2) + (\hat{a}_1 - \hat{d}_1)x(1) - \hat{a}_1 \hat{d}_1 x(0) \\
e(3) &= x(3) + \hat{a}_1 x(2) - \hat{d}_1 e(2) \\
&= x(3) + (\hat{a}_1 - \hat{d}_1)x(2) - (\hat{a}_1 - \hat{d}_1)\hat{d}_1 x(1) + \hat{a}_1 \hat{d}_1^2 x(0)
\end{aligned}$$

which shows that $e(n)$ is a nonlinear function of the model parameters if $Q \neq 0$. Thus, the total squared error

$$\varepsilon(\mathbf{c}) = \sum_{n=N_i}^{N_f} e^2(n) \tag{8.3.16}$$

expressed in terms of the signal values $x(0), x(1), \cdots, x(N-1)$, is a nonquadratic function of the model parameters. Sometimes, $\varepsilon(\mathbf{c})$ has several local minima. The model parameters can be obtained by minimizing the total square error using nonlinear optimization techniques.

8.4 Applications

Pole-zero modeling has many applications in such fields as spectral estimation, speech processing, geophysics, biomedical signal processing, and general time series analysis and forecasting (Marple 1987; Kay 1988; Robinson and Treitel 1980; Box, Jenkins, and Reinsel 1994). In this section, we discuss the application of pole-zero models to spectral estimation and speech processing.

8.4.1 Spectral Estimation

After we have estimated the parameters of a PZ model, we can compute the PSD of the analyzed process by

$$\hat{R}(e^{j\omega}) = \hat{\sigma}_\omega^2 \frac{\left|1+\sum_{k=1}^{Q}\hat{d}_k e^{j\omega k}\right|^2}{\left|1+\sum_{k=1}^{P}\hat{a}_k e^{j\omega k}\right|^2} \tag{8.4.1}$$

In practice, we mainly use AP models because (1) the all-zero PSD estimator is essentially identical to the Blackman-Tukey one (see Problem 8.16) and (2) the application of pole-zero PSD estimators is limited by computational and other practical difficulties. Also, any continuous PSD can be approximated arbitrarily well by the PSD of an AP(P) model if P is chosen large enough (Anderson 1971). However, in practice, the value of P is limited by the amount of available data (usually $P < N/3$). The statistical properties of all-pole PSD estimators are difficult to obtain; however, it has been shown that the estimator is consistent only if the analyzed process is AR(P_0) with $P_0 \le P$. Furthermore, the quality of the estimator degrades if the process is contaminated by noise. More details about pole-zero PSD estimation can be found in Kay (1988), Porat (1994), and Percival and Walden (1993).

The performance of all-pole PSD estimators depends on the method used to estimate the model parameters, the order of the model, and the presence of noise. The effect of model mismatch is shown in Figure 8.13 and is further investigated in Problem 8.17 Order selection in all-pole PSD estimation is absolutely critical: If P is too large, the obtained PSD exhibits spurious peaks; if P is too small, the structure of the PSD is smoothed over. The increased resolution of the parametric techniques, compared to the nonparametric PSD estimation methods, is basically the result of imposing structure on the data (i.e., a model). The model makes possible the extrapolation of the ACS, which in turns leads to better resolution. However, if the adopted model is inaccurate, that is, if it does not match the data, then the "gained" resolution reflects the model and not the data! As a result, despite their popularity and their "success" with simulated signals, the application of parametric PSD estimation techniques to actual experimental data is rather limited.

Figure 8.13 shows the results of a Monte Carlo simulation of various all-pole PSD estimation techniques. We see that, except for the windowing approach that results in a significant loss of resolution, all other techniques have similar performance. However, we should mention that the forward/backward LS all-pole modeling method is considered to provide the best results (Marple 1987).

In practice, it is our experience that the best way to estimate the PSD of an actual signal is to combine parametric *prewhitening* with nonparametric PSD estimation methods. The process is illustrated in Figure 8.14 and involves the following steps:

1. Fit an AP(P) model to the data using the forward LS, forward/backward LS, or Burg's method with no windowing.
2. Compute the residual (prediction error)

$$e(n) = x(n) + \sum_{k=1}^{P} a_k^* x(n-k) \qquad P \le n \le N-1 \tag{8.4.2}$$

   and then compute and plot its ACS, PACS, and cumulative periodogram (see Figure 8.2) to see if it is reasonably white. The goal is not to completely whiten the residual but to reduce its spectral dynamic range, that is, to increase its spectral flatness to *avoid* spectral leakage.
3. Compute the PSD $\hat{R}_e(e^{j\omega k})$, using one of the nonparametric techniques discussed in Chapter 4.

4. Compute the PSD of $x(n)$ by

$$\hat{R}_x(e^{j\omega_k}) = \frac{\hat{R}_e(e^{j\omega_k})}{|A(e^{j\omega_k})|^2} \tag{8.4.3}$$

that is, by applying postcoloring to "undo" the prewhitening.

The main goal of AP modeling here is to reduce the spectral dynamic range to avoid leakage. In other words, we need a good linear predictor regardless of whether the process is true AR(P). Therefore, very accurate order selection and model fit are not critical, because all spectral structure not captured by the model is still in the residuals. Needless to say, if the periodogram of $x(n)$ has a small dynamic range, we do not need prewhitening. Another interesting application of prewhitening is for the detection of outliers in practical data (Martin and Thomson 1982).

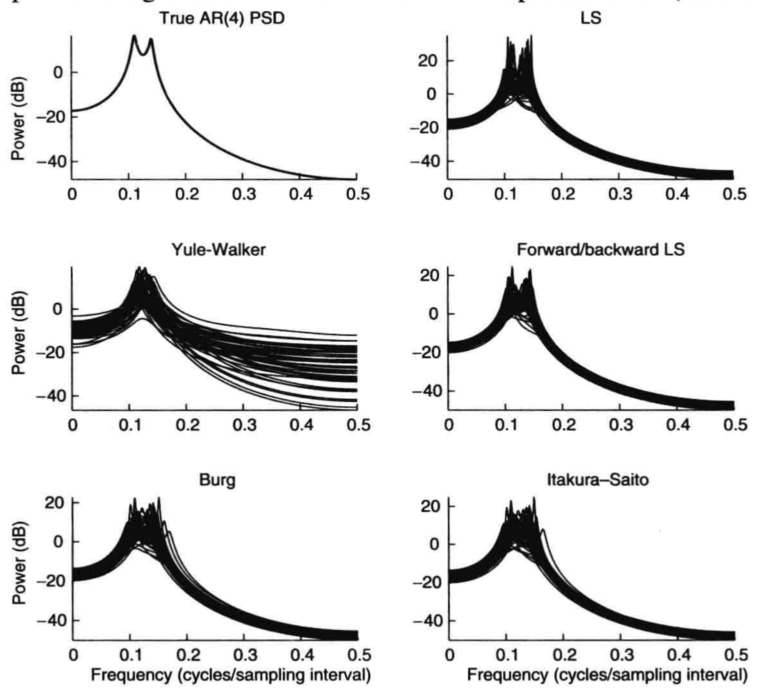


FIGURE 8.13
Monte Carlo simulation for the comparison of all-pole PSD estimation techniques, using 50 realizations of a 50-sample segment from an AR(4) process using fourth-order AP models.

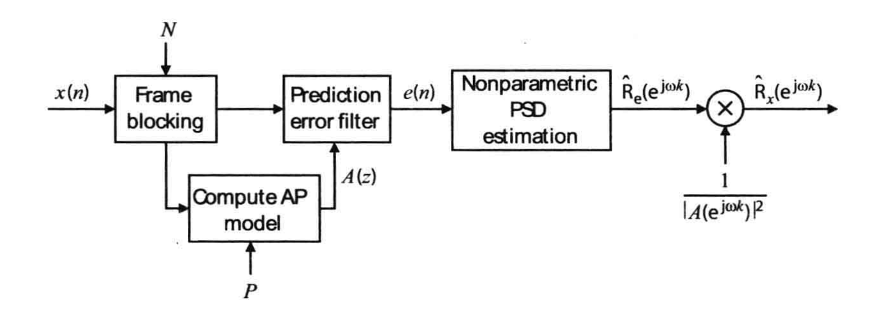


FIGURE 8.14
Block diagram of nonparametric PSD estimation using linear prediction prewhitening.

EXAMPLE 8.4.1. To illustrate the effectiveness of the above prewhitening and postcoloring method, consider the AR(4) process $x(n)$ used in Example 8.2.3. This process has a large dynamic range, and hence the nonparametric methods such as Welch's periodogram averaging method will suffer from leakage problems. Using the system function of the model

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - 2.7607z^{-1} + 3.8106z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

and WGN (0,1) input sequence, we generated 256 samples of $x(n)$. These samples were then used to obtain the all-pole LS predictor coefficients using the `arwin` function. The spectrum $|A(e^{j\omega})|^{-2}$ corresponding to this estimated model is shown in Figure 8.15 as a dashed curve. The signal samples were prewhitened using the model to obtain the residuals $e(n)$. The nonparametric PSD estimate $\hat{R}_e(e^{j\omega})$ of $e(n)$ was computed by using Welch's method with $L = 64$ and 50 percent overlap. Finally, $\hat{R}_e(e^{j\omega})$ was postcolored using the spectrum $|A(e^{j\omega})|^{-2}$ to obtain $\hat{R}_x(e^{j\omega})$, which is shown in Figure 8.16 as a solid line. For comparison purposes, the Welch PSD estimate of $x(n)$ is also shown as a dotted line. As expected, the nonparametric estimate does not resolve the two peaks in the true spectrum and suffers from leakage at high frequencies. However, the combined nonparametric and parametric estimate resolves two peaks with ease and also follows the true spectrum quite well. Therefore, the use of the parametric method as a preprocessor is highly recommended especially in large-dynamic-range situations.

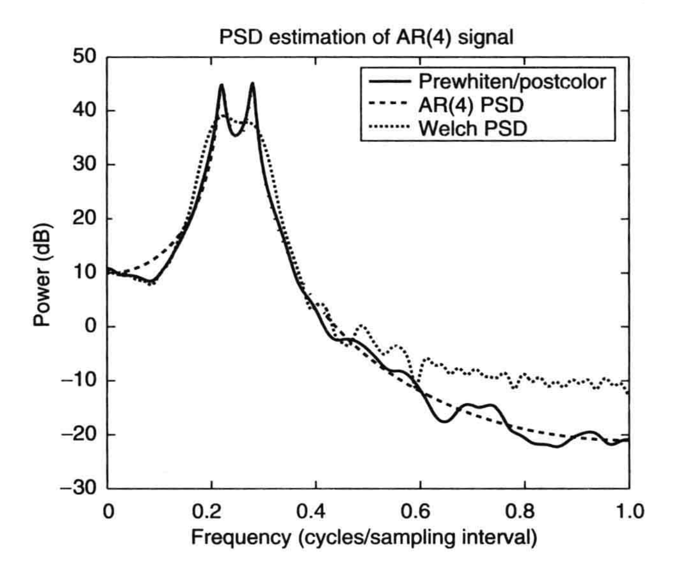


FIGURE 8.15
Spectral estimation of AR(4) process using prewhitening and postcoloring method in Example 8.4.1

8.4.2 *Speech Modeling*

All-pole modeling using LS linear prediction is widely employed in speech processing applications because (1) it provides a good approximation to the vocal tract for voiced sounds and adequate approximation for unvoiced and transient sounds, (2) it results in a good separation between source (fine spectral structure) and vocal tract (spectral envelop), and (3) it is analytically tractable and leads to efficient software and hardware implementations.

Figure 8.16 shows a typical AP modeling system, also known as the *linear predictive coding (LPC)* processor, that is used in speech synthesis, coding, and recognition applications. The processor operates in a block processing mode; that is, it processes a frame of N samples and computes a vector of model parameters using the following basic steps:

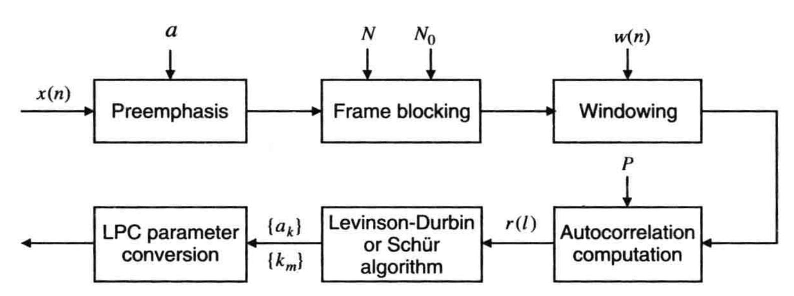


FIGURE 8.16
Block diagram of an AP modeling processor for speech coding and recognition.

1. *Preemphasis.* The digitized speech signal is filtered by the high-pass filter

$$H_1(z) = 1 - \alpha z^{-1} \quad 0.9 \le \alpha \le 1 \tag{8.4.4}$$

to reduce the dynamic range of the spectrum, that is, to flatten the spectral envelope, and make subsequent processing less sensitive to numerical problems (Makhoul 1975a). Usually $\alpha = 0.95$, which results in about a 32 dB boost in the spectrum at $\omega = \pi$ over that at $\omega = 0$. The preemphasizer can be made adaptive by setting $\alpha = \rho(1)$, where $\rho(l)$ is the normalized autocorrelation of the frame, which corresponds to a first-order optimum prediction error filter.
2. *Frame blocking.* Here the preemphasized signal is blocked into frames of N samples with successive frames overlapping by $N_0 = N/3$ samples. In speech recognition $N = 300$ with a sampling rate $F_s = 6.67$ Hz, which corresponds to 45-ms frames overlapping by 15 ms.
3. *Windowing.* Each frame is multiplied by an N-sample window (usually Hamming) to smooth the discontinuities at the beginning and the end of the frame.
4. *Autocorrelation computation.* Here the LPC processor computes the first $P+1$ values of the autocorrelation sequence. Usually, $P = 8$ in speech recognition and $P = 12$ in speech coding applications. The value of $r(0)$ provides the energy of the frame, which is useful for speech detection.
5. *LPC analysis.* In this step the processor uses the $P+1$ autocorrelations to compute an LPC parameter set for each speech frame. Depending on the required parameters, we can use the algorithm of Levinson-Durbin or the algorithm of Schür. The most widely used parameters are

$$a_m = a_m^{(P)} \qquad \text{LPC coefficients}$$

$$k_m \qquad \text{PACS}$$

$$g_m = \frac{1}{2}\log\frac{1-k_m}{1+k_m} = \tanh^{-1} k_m \qquad \text{log area ratio coefficients}$$

$$c(m) \qquad \text{cepstral coefficients}$$

$$\omega_m \qquad \text{line spectrum pairs}$$

where $1 \le m \le P$, except for the cepstrum, which is computed up to about $3P/2$. The line spectrum pair parameters, and their application to speech processing is considered in Furui (1989).

The log area ratio and the line spectrum pair coefficients have good quantization properties and are used for speech coding (Rabiner and Schafer 1978; Furui 1989); the cepstral coefficients provide an excellent discriminant for speech and speaker recognition applications (Rabiner and Juang 1993; Mammone et al. 1996). AP models are extensively used for the modeling of speech sounds. However, the AP model does not provide an accurate description of the speech spectral envelope when the speech production process resembles a PZ system (Atal and Schroeder 1978). This can happen when (1) the nasal tract is coupled to the main vocal tract through the velar opening, for example, during the generation of nasals and nasalized sounds, (2) the source of excitation is not at the glottis but is in the interior of the vocal tract (Flanagan 1972), and (3) the transmission or recording channel has zeros in its response. Although a zero can be approximated with arbitrary precision by a number of poles, this approximation is usually inefficient and leads to spectral distortion and other problems. These problems can be avoided by using pole-zero modeling, as illustrated in the following example. More details about pole-zero speech modeling can be found in Atal and Schroeder (1978).

Figure 8.17(*a*) shows a Hamming window segment from an artificial nasal speech signal sampled at F_s = 10 kHz. According to acoustic theory, such sounds require both poles and zeros in the vocal tract system function. Before the fitting of the model, the data are passed though a preemphasis filter with $\alpha = 0.95$. Figure 8.18(*b*) shows the periodogram of the speech segment, the spectrum of an AP(16) model using data windowing, and the spectrum of a PZ(12, 6) model using the least-squares algorithm. We see that the pole-zero model matches zeros ("valleys") in the periodogram of the data better than other models do.

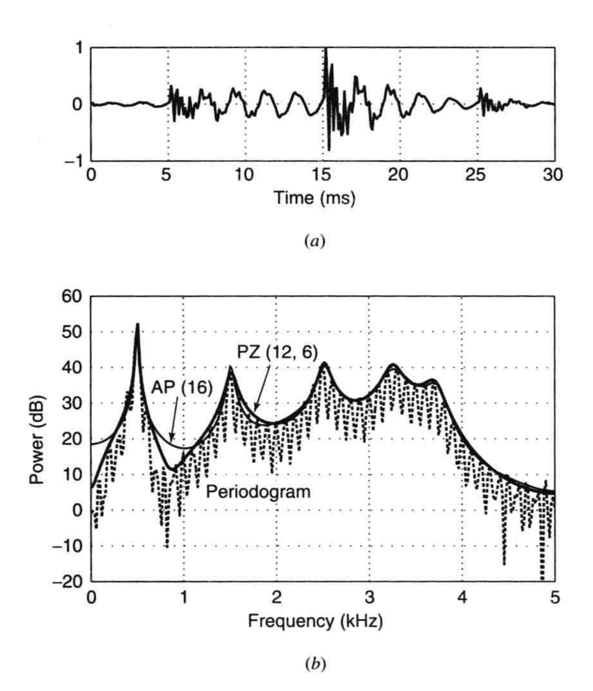


FIGURE 8.17
(*a*) Speech segment and; (*b*) periodogram, spectrum of a data windowing-based AP(16) model, and spectrum of a residual windowing-based PZ(12, 6)model.

8.5 Harmonic Models and Frequency Estimation Techniques

The pole-zero models we have discussed so far assume a linear time-invariant system that is excited by white noise. However, in many applications, the signals of interest are complex exponentials contained in white noise for which a *sinusoidal* or *harmonic model* is more appropriate. Signals consisting of complex exponentials are found as formant frequencies in speech processing, moving targets in radar, and spatially propagating signals in array processing.[2] For real signals, complex exponentials make up a complex conjugate pair (sinusoids), whereas for complex signals, they may occur at a single frequency.

For complex exponentials found in noise, the parameters of interest are the frequencies of the signals. Therefore, our goal is to estimate these frequencies from the data. One might consider estimating the power spectrum by using the nonparametric methods discussed in Chapter 4. The frequency estimates of the complex exponentials are then the frequencies at which peaks occur in the spectrum. Certainly, the use of these nonparametric methods seems appropriate for complex exponential signals since they make no assumptions about the underlying process. We might also consider making use of an all-pole model for the purposes of spectrum estimation as discussed in Section 8.4.1, also known as the maximum entropy method (MEM) spectral estimation technique. Even though some of these methods can achieve very fine resolution, none of these methods accounts for the underlying model of complex exponentials in noise. As in all modeling problems, the use of the appropriate model is desirable from an intuitive point of view and advantageous in terms of performance. We begin by describing the harmonic signal model, deriving the model in a vector notation, and looking at the eigendecomposition of the correlation matrix of complex exponentials in noise. Then we describe frequency estimation methods based on the harmonic model: the Pisarenko harmonic decomposition, and the MUSIC, minimum-norm, and ESPRIT algorithms.

These methods have the ability to resolve complex exponentials closely spaced in frequency and has led to the name *superresolution* commonly being associated with them. However, a word of caution on the use of these

[2]In array processing, a spatially propagating wave produces a complex exponential signal as measured across uniformly spaced sensors in an array. The frequency of the complex exponential is determined by the angle of arrival of the impinging, spatially propagating signal. Thus, in array processing the frequency estimation problem is known as *angle-of-arrival (AOA)* or *direction-of-arrival (DOA)* estimation.

harmonic models. The high level of performance in terms of resolution is achieved by assuming an underlying model of the data. As with all other parametric methods, the performance of these techniques depends upon how closely this mathematical model matches the actual physical process that produced the signals. Deviations from this assumption result in model mismatch and will produce frequency estimates for a signal that may not have been produced by complex exponentials. In this case, the frequency estimates have little meaning.

8.5.1 Harmonic Model

Consider the signal model that consists of P complex exponentials in noise

$$x(n) = \sum_{p=1}^{P} \alpha_p \, \mathrm{e}^{\mathrm{j}2\pi n f_p} + \omega(n) \tag{8.5.1}$$

The normalized, discrete-time frequency of the p th component is

$$f_p = \frac{\omega_p}{2\pi} = \frac{F_p}{F_s} \tag{8.5.2}$$

where ω_p is the discrete-time frequency in radians, F_p is the actual frequency of the pth complex exponential, and F_s is the sampling frequency. The complex exponentials may occur either individually or in complex conjugate pairs, as in the case of real signals. In general, we want to estimate the frequencies and possibly also the amplitudes of these signals. Note that the phase of each complex exponential is contained in the amplitude, that is,

$$\alpha_p = |\alpha_p| \, \mathrm{e}^{\mathrm{j}\psi_p} \tag{8.5.3}$$

where the phases ψ_p are uncorrelated random variables uniformly distributed over $[0, 2\pi]$. The magnitude $|\alpha_p|$ and the frequency f_p are deterministic quantities. If we consider the spectrum of a harmonic process, we note that it consists of a set of impulses with a constant background level at the power of the white noise $\sigma_\omega^2 = E\{|\omega(n)|^2\}$. As a result, the power spectrum of complex exponentials is commonly referred to as a *line spectrum*, as illustrated in Figure 8.18.

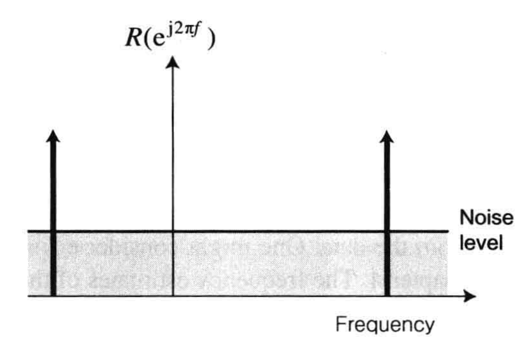


FIGURE 8.18
The spectrum of complex exponentials in noise

Since we will make use of matrix methods based on a certain time window of length M, it is useful to characterize the signal model in the form of a vector over this time window consisting of the sample delays of the signal. Consider the signal $x(n)$ from (8.5.1) at its current and future $M-1$ values. This time window can be written as

$$\boldsymbol{x}(n) = [x(n) x\ (n+1) \ \cdots \ x(n+M-1)]^{\mathrm{T}} \tag{8.5.4}$$

We can then write the signal model consisting of complex exponentials in noise from (8.5.1) for a length-M time-window vector as

$$\boldsymbol{x}(n) = \sum_{p=1}^{P} \alpha_p \boldsymbol{v}(f_p) \mathrm{e}^{\mathrm{j}2\pi n f_p} + \boldsymbol{w}(n) = \boldsymbol{s}(n) + \boldsymbol{w}(n) \tag{8.5.5}$$

where $\boldsymbol{w}(n) = [\omega(n) \ \omega(n+1) \ \cdots \ \omega(n+M-1)]^{\mathrm{T}}$ is the time-window vector of white noise and

$$\boldsymbol{v}(f)=[1\ \mathrm{e}^{\mathrm{j}2\pi f}\ \cdots\ \mathrm{e}^{\mathrm{j}2\pi(M-1)f}]^{\mathrm{T}} \tag{8.5.6}$$

is the time-window frequency vector. Note that $\boldsymbol{v}(f)$ is simply a length-M DFT vector at frequency f. We differentiate here between the signal $s(n)$, consisting of the sum of complex exponentials, and the noise component $\boldsymbol{w}(n)$, respectively.

Consider the time-window vector model consisting of a sum of complex exponentials in noise from (8.5.5). The autocorrelation matrix of this model can be written as the sum of signal and noise autocorrelation matrices

$$\begin{aligned}\boldsymbol{R}_x &= E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\}=\boldsymbol{R}_s+\boldsymbol{R}_\omega \\ &=\sum_{p=1}^{P}|\alpha_p|^2\,\boldsymbol{v}(f_p)\boldsymbol{v}^{\mathrm{H}}(f_p)+\sigma_\omega^2\boldsymbol{I}=\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{\mathrm{H}}+\sigma_\omega^2\boldsymbol{I}\end{aligned} \tag{8.5.7}$$

where
$$\boldsymbol{V}=[\,\boldsymbol{v}(f_1)\,\boldsymbol{v}(f_2)\ \cdots\ \boldsymbol{v}(f_P)) \tag{8.5.8}$$
is an $M\times P$ matrix whose columns are the time-window frequency vectors from (8.5.6) at frequencies f_p of the complex exponentials and

$$\boldsymbol{A}=\begin{bmatrix}|\alpha_1|^2 & 0 & \cdots & 0\\ 0 & |\alpha_2|^2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & |\alpha_P|^2\end{bmatrix} \tag{8.5.9}$$

is a diagonal matrix of the powers of each of the respective complex exponentials. The autocorrelation matrix of the white noise is

$$\boldsymbol{R}_\omega=\sigma_\omega^2\boldsymbol{I} \tag{8.5.10}$$

which is full rank, as opposed to $\boldsymbol{R}_s$ which is rank-deficient for $P<M$. In general, we will always choose the length of our time window M to be greater than the number of complex exponentials P.

The autocorrelation matrix can also be written in terms of its eigendecomposition

$$\boldsymbol{R}_x=\sum_{m=1}^{M}\lambda_m\boldsymbol{q}_m\boldsymbol{q}_m^{\mathrm{H}}=\boldsymbol{Q}\Lambda\boldsymbol{Q}^{\mathrm{H}} \tag{8.5.11}$$

where λ_m are the eigenvalues in descending order, that is, $\lambda_1\ge\lambda_2\ge\cdots\ge\lambda_M$, and $\mathbf{q}_m$ are their corresponding eigenvectors. Here Λ is a diagonal matrix made up of the eigenvalues found in descending order on the diagonal, while the columns of $\boldsymbol{Q}$ are the corresponding eigenvectors. The eigenvalues due to the signals can be written as the sum of the signal power in the time window and the noise:

$$\lambda_m=M\,|\alpha_m|^2+\sigma_\omega^2 \qquad \text{for}\quad m\le P \tag{8.5.12}$$

The remaining eigenvalues are due to the noise only, that is,

$$\lambda_m=\sigma_\omega^2 \qquad \text{for}\quad m>P \tag{8.5.13}$$

Therefore, the P largest eigenvalues correspond to the signal made up of complex exponentials and the remaining eigenvalues have equal value and correspond to the noise. Thus, we can partition the correlation matrix into portions due to the signal and noise eigen-vectors

$$\begin{aligned}\boldsymbol{R}_x&=\sum_{m=1}^{P}(M\,|\alpha_m|^2+\sigma_\omega^2)\boldsymbol{q}_m\boldsymbol{q}_m^{\mathrm{H}}+\sum_{m=P+1}^{M}\sigma_\omega^2\boldsymbol{q}_m\boldsymbol{q}_m^{\mathrm{H}}\\ &=\boldsymbol{Q}_s\Lambda_s\boldsymbol{Q}_s^{\mathrm{H}}+\sigma_\omega^2\boldsymbol{Q}_\omega\boldsymbol{Q}_\omega^{\mathrm{H}}\end{aligned} \tag{8.5.14}$$

where
$$\boldsymbol{Q}_s=[\boldsymbol{q}_1\ \boldsymbol{q}_2\ \cdots\ \boldsymbol{q}_P] \qquad \boldsymbol{Q}_\omega=[\boldsymbol{q}_{P+1}\ \cdots\ \boldsymbol{q}_M] \tag{8.5.15}$$

are matrices whose columms consist of the signal and noise eigenvectors, respectively. The matrix Λ_s is a $P \times P$ diagonal matrix containing the signal eigenvalues from (8.5.12). Thus, the M -dimensional subspace that contains the observations of the time-window signal vector from (8.5.5) can be split into two subspaces spanned by the signal and noise eigenvectors, respectively. These two subspaces, known as the *signal subspace* and the *noise subspace*, are orthogonal to each other since the correlation matrix is Hermitian symmetric.[3] All the subspace methods discussed later in this section rely on the partitioning of the vector space into signal and noise subspaces. Recall from Chapter 7 in (7.2.29) that the projection matrix from an M -dimensional space onto an L-dimensional subspace ($L < M$) spanned by a set of vectors $\boldsymbol{Z} = [z_1 z_2 \cdots z_L]$ is

$$\boldsymbol{P} = \boldsymbol{Z}(\boldsymbol{Z}^{\mathrm{H}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{H}} \tag{8.5.16}$$

Therefore, we can write the matrices that project an arbitrary vector onto the signal and noise subspaces as

$$\boldsymbol{P}_s = \boldsymbol{Q}_s\boldsymbol{Q}_s^{\mathrm{H}} \qquad \boldsymbol{P}_\omega = \boldsymbol{Q}_\omega\boldsymbol{Q}_\omega^{\mathrm{H}} \tag{8.5.17}$$

since the eigenvectors of the correlation matrix are orthonormal ($\boldsymbol{Q}_s^H\boldsymbol{Q}_s = \boldsymbol{I}$ and $\boldsymbol{Q}_\omega^H\boldsymbol{Q}_\omega = \boldsymbol{I}$). Since the two subspaces are orthogonal

$$\boldsymbol{P}_\omega\boldsymbol{Q}_s = \boldsymbol{0} \qquad \boldsymbol{P}_s\boldsymbol{Q}_\omega = \boldsymbol{0} \tag{8.5.18}$$

then all the time-window frequency vectors from (8.5.5) must lie completely in the signal subspace, that is,

$$\boldsymbol{P}_s\boldsymbol{v}(f_p) = \boldsymbol{v}(f_p) \qquad \boldsymbol{P}_\omega\boldsymbol{v}(f_p) = \boldsymbol{0} \tag{8.5.19}$$

These concepts are central to the subspace-based frequency estimation methods discussed in Sections 8.6.2 through 8.6.5.

Note that in our analysis, we are considering the theoretical or true correlation matrix $\boldsymbol{R}_x$. In practice, the correlation matrix is not known and must be estimated from the measured data samples. If we have a time-window signal vector from (8.5.4), then we can form the data matrix by stacking the rows with measurements of the time-window data vector at a time n

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}(0) \\ \boldsymbol{x}^{\mathrm{T}}(1) \\ \vdots \\ \boldsymbol{x}^{\mathrm{T}}(n) \\ \vdots \\ \boldsymbol{x}^{\mathrm{T}}(N-2) \\ \boldsymbol{x}^{\mathrm{T}}(N-1) \end{bmatrix} = \begin{bmatrix} x(0) & x(1) & \cdots & x(M-1) \\ x(1) & x(2) & \cdots & x(M) \\ \vdots & \vdots & \vdots & \vdots \\ x(n) & x(n+1) & \cdots & x(n+M-1) \\ \vdots & \vdots & \vdots & \vdots \\ x(N-2) & x(N-1) & \cdots & x(N+M-3) \\ x(N-1) & x(N) & \cdots & x(N+M-2) \end{bmatrix} \tag{8.5.20}$$

which has dimensions of $N \times M$, where N is the number of data records or snapshots and M is the time-window length. From this matrix, we can form an estimate of the correlation matrix, referred to as the sample correlation matrix

$$\hat{\boldsymbol{R}}_x = \frac{1}{N}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X} \tag{8.5.21}$$

In the case of an estimated sample correlation matrix, the noise eigenvalues are no longer equal because of the finite number of samples used to compute $\hat{\boldsymbol{R}}$. Therefore, the nice, clean threshold between signal and noise eigenvalues, as described in (8.5.12) and (8.5.13), no longer exists. The model order estimation techniques discussed in Section 8.2 can be employed to attempt to determine the number of complex exponentials P present. In practice, these methods are best used as rough estimates, as their performance is not very accurate, especially for short data records.

For several of the frequency estimation techniques described in this section, the analysis considers the use of

---

[3]The eigenvectors of a Hermitian symmetric matrix are orthogonal.

eigenvalues and eigenvectors of the correlation matrix for the purposes of defining signal and noise subspaces.[4] In practice, we estimate the signal and noise subspaces by using the eigenvectors and eigenvalues of the sample correlation matrix. Note that for notational expedience we will not differentiate between eigenvectors and eigenvalues of the true and sample correlation matrices. However, the reader should always keep in mind that the sample correlation matrix eigendecomposition is what must be used for implementation. We note that use of an estimate rather than the true correlation matrix will result in a degradation in performance, the analysis of which is beyond the scope of this book.

8.5.2 Pisarenko Harmonic Decomposition

The *Pisarenko harmonic decomposition* (PHD) was the first frequency estimation method proposed that was based on the eigendecomposition of the correlation matrix and its partitioning into signal and noise subspaces (Pisarenko 1973). This method uses the eigenvector associated with the smallest eigenvalue to estimate the frequencies of the complex exponentials. Although this method has limited practical use owing to its sensitivity to noise, it is of great theoretical interest because it was the first method based on signal and noise subspace principles and it helped to fuel the development of many well-known subspace methods, such as MUSIC and ESPRIT.

Consider the model of complex exponentials contained in noise in (8.5.5) and the eigendecomposition of its correlation matrix in (8.5.14). The eigenvector corresponding to the minimum eigenvalue must be orthogonal to all the eigenvectors in the signal subspace. Thus, we choose the time window to be of length

$$M = P + 1 \tag{8.5.22}$$

that is, 1 greater than the number of complex exponentials. Therefore, the noise subspace consists of a single eigenvector

$$\mathbf{Q}_{\omega} = \mathbf{q}_M \tag{8.5.23}$$

corresponding to the minimum eigenvalue λ_M. By virtue of the orthogonality between the signal and noise subspaces, each of the P complex exponentials in the time-window signal vector model in (8.5.5) is orthogonal to this eigenvector

$$\mathbf{v}^{\mathrm{H}}(f_p)\mathbf{q}_M = \sum_{k=1}^{M} q_M(k)\mathrm{e}^{-\mathrm{j}2\pi f_p(k-1)} = 0 \qquad \text{for} \quad m \le P \tag{8.5.24}$$

Making use of this property, we can compute

$$\bar{R}_{\mathrm{phd}}(\mathrm{e}^{\mathrm{j}2\pi f}) = \frac{1}{|\mathbf{v}^{\mathrm{H}}(f)\mathbf{q}_M|^2} = \frac{1}{|Q_M(\mathrm{e}^{\mathrm{j}2\pi f})|^2} \tag{8.5.25}$$

which is commonly referred to as a *pseudospectrum*. The frequencies are then estimated by observing the P peaks in $\bar{R}_{\mathrm{phd}}(e^{j2\pi f})$. Note that since (8.5.25) requires a search of all frequencies $-0.5 \le f \le 0.5$, in practice a dense sampling of the frequencies is generally necessary. The quantity

$$Q_M(\mathrm{e}^{\mathrm{j}2\pi f}) = \mathbf{v}^{\mathrm{H}}(f)\mathbf{q}_M = \sum_{k=1}^{M} q_M(k)\mathrm{e}^{-\mathrm{j}2\pi f(k-1)} \tag{8.5.26}$$

is simply the Fourier transform of the Mth eigenvector corresponding to the minimum eigenvalue. Thus, the pseudospectrum for the Pisarenko harmonic decomposition $\bar{R}_{\mathrm{phd}}(\mathrm{e}^{\mathrm{j}2\pi f})$ can be efficiently implemented by computing the FFT of $\mathbf{q}_M$ with sufficient zero padding to provide the necessary frequency resolution. Then $\bar{R}_{\mathrm{phd}}(\mathrm{e}^{\mathrm{j}2\pi f})$ is simply the reciprocal of the spectrum of the noise eigenvector, that is, the squared magnitude of its Fourier transform. Note that $\bar{R}_{\mathrm{phd}}(\mathrm{e}^{\mathrm{j}2\pi f})$ is not an estimate of the true power spectrum since it contains no information about the powers of the complex exponentials $|\alpha_p|^2$ or the background noise level σ_{ω}^2. However, these amplitudes can be found by using the estimated frequencies and the corresponding time-window frequency vectors along with the relationship of eigenvalues and eigenvectors. See Problem 8.24 for details.

Alternately, the frequencies of the complex exponentials can be found by computing the zeros of the Fourier

[4] The ESPRIT method uses a singular value decomposition of data matrix **X**.

transform of the Mth eigenvector in (8.5.23). The z-transform of this eigenvector is

$$Q_M(z) = \sum_{k=1}^{M} q_M(k) z^{-k} = \prod_{k=1}^{M-1} (1 - e^{j2\pi f_k} z^{-1}) \tag{8.5.27}$$

where the phases of the $P = M - 1$ roots of this polynomial are the frequencies f_k of the $P = M - 1$ complex exponentials.

As we stated up front, the significance of the Pisarenko harmonic decomposition is seen mostly from a theoretical perspective. The limitations of its practical use stem from the fact that it uses a single noise eigenvector and, as a result, lacks the necessary robustness needed for most applications. Since the correlation matrix is not known and must be estimated from data, the resulting noise eigenvector of the estimated correlation matrix is only an estimate of the actual noise eigenvector. Because we only use one noise eigenvector, this method is very sensitive to any errors in the estimation of the noise eigenvector.

EXAMPLE 8.5.1 We demonstrate the use of the Pisarenko harmonic decomposition with a sinusoid in noise. The amplitude and frequency of the sinusoid are $\alpha = 1$ and $f = 0.2$, respectively. The additive noise has unit power ($\sigma_\omega^2 = 1$). Using MATLAB, this signal is generated:

```
x=sin(2*pi*f*[0:N-1]]')+(randn(n,1)+j*randn(N,1))/sqrt(2);
```

Since the number of complex exponentials is equal to $P = 2$ (a complex conjugate pair for a sinusoid), the time-window length is chosen to be $M = 3$. After forming the $N \times M$ data matrix $\mathbf{X}$ and computing the sample correlation matrix $\hat{\mathbf{R}}_x$, we can compute the pseudospectrum as follows:

```
[Q0, D]=eig(R); %eigendecomposition
[lambda, indes]=sort(abs(diag(d))); %order by eigenvalue magnitude
lambda=lambda(M:-1:1); Q=Q0(:,index(M:-1:1));
Rbar=1./abs(fftshift(fft(Q(:,M),Nfft))).^2;
```

Figure 8.19 shows the pseudospectrum of the Pisarenko harmonic decomposition for a single realization with an FFT size of 1024. Note the two peaks near $f = \pm 0.2$. Recall that this is a pseudospectrum, so that the actual values do not correspond to an estimate of power. A MATLAB routine for estimating frequencies using the Pisarenko harmonic decomposition is provided in `phd.m`.

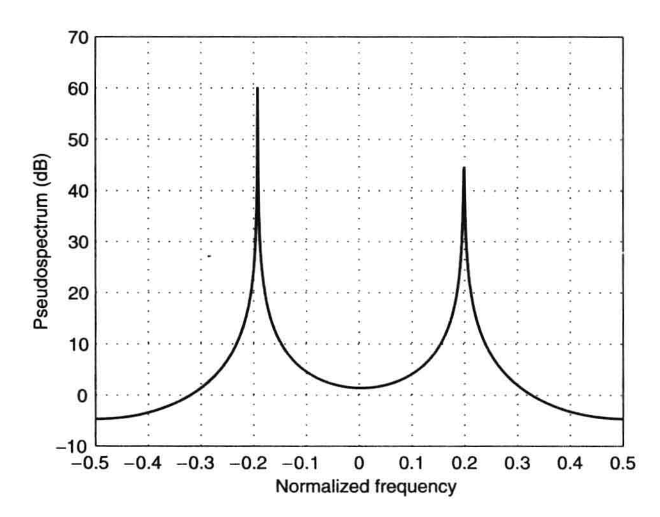


FIGURE 8.19
Pseudospectrum for the Pisarenko harmonic decomposition of a sinusoid in noise with frequency $f = 0.2$

8.5.3 MUSIC Algorithm

The *multiple signal classification (MUSIC)* frequency estimation method was proposed as an improvement on the Pisarenko harmonic decomposition (Bienvenu and Kopp 1983; Schmidt 1986). Like the Pisarenko harmonic decomposition, the M-dimensional space is split into signal and noise components using the eigenvectors of the correlation matrix from (8.5.15). However, rather than limit the length of the time window to $M = P + 1$, that is, 1

greater than the number of complex exponentials, allow the size of the time window to be $M > P+1$. Therefore, the noise subspace has a dimension greater than 1. Using this larger dimension allows for averaging over the noise subspace, providing an improved, more robust frequency estimation method than Pisarenko harmonic decomposition.

Because of the orthogonality between the noise and signal subspaces, all the time-window frequency vectors of the complex exponentials are orthogonal to the noise subspace from (8.5.19). Thus, for each eigenvector ($P < m \leq M$)

$$\boldsymbol{v}^{\mathrm{H}}(f_p)\boldsymbol{q}_m = \sum_{k=1}^{M} q_m(k)\mathrm{e}^{-\mathrm{j}2\pi f_p(k-1)} = 0 \tag{8.5.28}$$

for all the P frequencies f_p of the complex exponentials. Therefore, if we compute a pseudospectrum for each noise eigenvector as

$$\overline{R}_m(\mathrm{e}^{\mathrm{j}2\pi f}) = \frac{1}{|\boldsymbol{v}^{\mathrm{H}}(f)\boldsymbol{q}_m|^2} = \frac{1}{|Q_m(\mathrm{e}^{\mathrm{j}2\pi f})|^2} \tag{8.5.29}$$

the polynomial $Q_m(\mathrm{e}^{\mathrm{j}2\pi f})$ has $M-1$ roots, P of which correspond to the frequencies of the complex exponentials. These roots produce P peaks in the pseudospectrum from (8.5.29). Note that the pseudospectra of all $M-P$ noise eigenvectors share these roots that are due to the signal subspace. The remaining roots of the noise eigenvectors, however, occur at different frequencies. There are no constraints on the location of these roots, so that some may be close to the unit circle and produce extra peaks in the pseudospectrum. A means of reducing the levels of these spurious peaks in the pseudospectrum is to average the $M-P$ pseudospectra of the individual noise eigenvectors

$$\overline{R}_{\text{music}}(\mathrm{e}^{\mathrm{j}2\pi f}) = \frac{1}{\sum_{m=P+1}^{M} |\boldsymbol{v}^{\mathrm{H}}(f)\boldsymbol{q}_m|^2} = \frac{1}{\sum_{m=P+1}^{M} |Q_m(\mathrm{e}^{\mathrm{j}2\pi f})|^2} \tag{8.5.30}$$

which is known as the MUSIC pseudospectrum. The frequency estimates of the P complex exponentials are then taken as the P peaks in this pseudospectrum. Again, the term *pseudospectrum* is used because the quantity in (8.5.30) does not contain information about the powers of the complex exponentials or the background noise level. Note that for $M = P+1$, the MUSIC method is equivalent to Pisarenko harmonic decomposition.

The implicit assumption in the MUSIC pseudospectrum is that the noise eigenvalues all have equal power $\lambda_m = \sigma_\omega^2$, that is, the noise is white. However, in practice, when an estimate is used in place of the actual correlation matrix, the noise eigenvalues will not be equal. The differences become more pronounced when the correlation matrix is estimated from a small number of data samples. Thus, a slight variation on the MUSIC algorithm, known as the *eigenvector (ev) method*, was proposed to account for the potentially different noise eigenvalues (Johnson and DeGraaf 1982). For this method, the pseudospectrum is

$$\overline{R}_{\text{ev}}(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{\sum_{m=P+1}^{M} \frac{1}{\lambda_m}|\boldsymbol{v}^{\mathrm{H}}(f)\boldsymbol{q}_m|^2} = \frac{1}{\sum_{k=P+1}^{M} \frac{1}{\lambda_m}|Q_m(\mathrm{e}^{\mathrm{j}2\pi f})|^2} \tag{8.5.31}$$

where λ_m is the eigenvalue corresponding to the eigenvector $\mathbf{q}_m$. The pseudospectrum of each eigenvector is normalized by its corresponding eigenvalue. In the case of equal noise eigenvalues ($\lambda_m = \sigma_\omega^2$,) for $P+1 \leq m \leq M$, the eigenvector and MUSIC methods are identical.

The peaks in the MUSIC pseudospectrum correspond to the frequencies at which the denominator in (8.5.30) $\sum_{m=P+1}^{M} |Q_m(\mathrm{e}^{\mathrm{j}2\pi f})|^2$ approaches zero. Therefore, we might want to consider the z-transform of this denominator

$$\overline{P}_{\text{music}}(z) = \sum_{m=P+1}^{M} Q_m(z)Q_m^*\left(\frac{1}{z^*}\right) \tag{8.5.32}$$

which is the sum of the z-transforms of the pseudospectrum due to each noise eigenvector. This $(2M-1)$ th-order polynomial has $M-1$ pairs of roots with one inside and one outside the unit circle. Since we assume that the complex exponentials are not damped, their corresponding roots must lie on the unit circle. Thus, if we have found the $M-1$ roots of (8.5.32), the P closest roots to the unit circle will correspond to the complex exponentials. The phases of these roots are then the frequency estimates. This method of rooting the polynomial corresponding to the MUSIC pseudospectrum is known as *root-MUSIC* (Barabell 1983). Note that in many cases, a rooting method is more efficient than computing a pseudospectrum at a very fine frequency resolution that may require a very large

FFT. Statistical performance analyses of the MUSIC algorithm can be found in Kaveh and Barabell (1986) and Stoica and Nehorai (1989). For the performance of the root-MUSIC method see Rao and Hari (1989). A routine for the MUSIC algorithm is provided in `music.m` and a routine for the root-MUSIC algorithm is provided in `rootmusic.m`.

EXAMPLE 8.5.2. In this example, we demonstrate the use of the MUSIC algorithm and examine its performance in terms of resolution with respect to that of the minimum-variance spectral estimator. Consider the following scenario: Two complex exponentials in unit power noise ($\sigma_\omega^2 = 1$) with normalized frequencies $f = 0.1,\ 0.2$ both with amplitudes of $\alpha = 1$. We generate $N = 128$ samples of the signal and use a frequency vector of length $M = 8$. Proceeding as we did in Example 8.5.1, we compute the eigendecomposition and partition it into signal and noise subspaces. The MUSIC pseudospectrum is computed as

```
Qbar=zeros(Nfft, 1);
for n=1:(M-P)
    Qbar=Qbar+abs(fftshift(fft(Q(:,M-(n-1)),Nfft))).^2;
end
Rbar=1./Qbar;
```

The minimum-variance spectral estimate and the MUSIC pseudospectrum are computed and averaged over 1000 realizations using an FFT size of 1024. The result is shown in Figure 8.20. The two exponentials have been clearly resolved using the MUSIC algorithm, whereas they are not very clear using the minimum-variance spectral estimate. Since the minimum-variance spectral estimator is nonparametric and makes no assumptions about the underlying model, it cannot achieve the resolution of the MUSIC algorithm.

8.5.4 *Minimum-Norm Method*

The minimum-norm method (Kumaresan and Tufts 1983), like the MUSIC algorithm, uses a time-window vector of length $M > P+1$ for the purposes of frequency estimation. For MUSIC, a larger time window is used than for Pisarenko harmonic decomposition, resulting in a larger noise subspace. The use of a larger subspace provides the necessary robustness for frequency estimation when an estimated correlation matrix is used. The same principle is applied in the minimum-norm frequency estimation method. However, rather than average the pseudospectra of all the noise subspace eigenvectors to reduce spurious peaks, as in the case of the MUSIC algorithm, a different approach is taken.

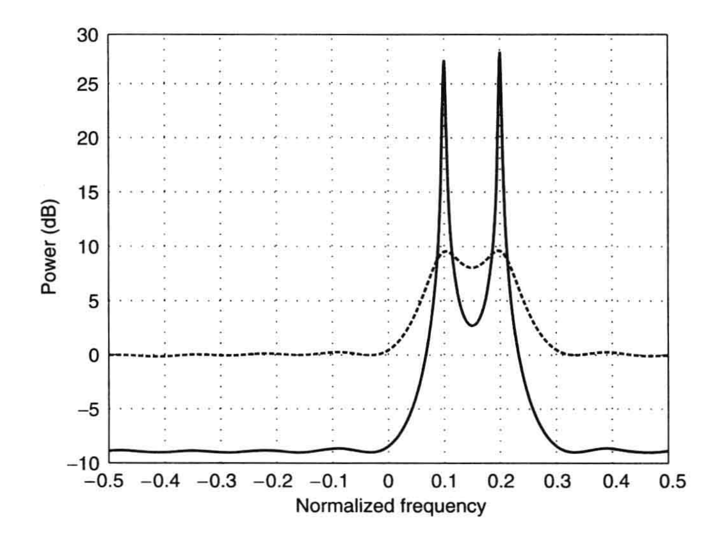


FIGURE 8.20
Comparison of the minimum-variance spectral estimate (dashed line) and the MUSIC pseudospectrum (solid line) for two complex exponentials in noise.

Consider a single vector $\boldsymbol{u}$ contained in the noise subspace. The pseudospectrum of this vector is given by

$$\bar{R}(e^{j2\pi f}) = \frac{1}{|v^H(f)u|^2} \tag{8.5.33}$$

Since the vector u lies in the noise subspace, its pseudospectrum in (8.5.33) has P peaks corresponding to the complex exponentials in the signal subspace. However, u is length M so that its pseudospectrum may exhibit an additional $M-P-1$ peaks that do not correspond to the frequencies of the complex exponentials. These spurious peaks lead to frequency estimation errors. In the case of Pisarenko harmonic decomposition, spurious peaks were not a concern since $M = P+1$ and therefore its pseudospectrum in (8.5.25) only had P peaks. On the other hand, the MUSIC algorithm diluted the strength of these spurious peaks since its pseudospectrum in (8.5.30) is produced by averaging the pseudospectra of the $M-P$ noise eigenvectors.

Recall the projection onto the noise subspace from (8.6.17) is

$$P_\omega = Q_\omega Q_\omega^H \tag{8.5.34}$$

where Q_ω is the matrix of noise eigenvectors. Therefore, for any vector u that lies in the noise subspace

$$P_\omega u = u \qquad P_s u = 0 \tag{8.5.35}$$

where P_s is the signal subspace projection matrix and $\mathbf{0}$ is the length-P zero vector. Now let us consider the z-transform of the coefficients of $u = [u(1)u(2)\cdots u(M)]^T$

$$U(z) = \sum_{k=0}^{M-1} u(k1)z^{-k} = \prod_{k=1}^{P}(1-e^{j2\pi f_k}z^{-1})\prod_{k=P+1}^{M-1}(1-z_k z^{-1}) \tag{8.5.36}$$

This polynomial is the product of the P roots corresponding to complex exponentials that lie on the unit circle and the $M-P-1$ roots that in general do not lie directly on the unit circle but can potentially produce spurious peaks in the pseudospectrum of u. Therefore, we want to choose u so that it minimizes the spurious peaks due to these other roots of its associated polynomial $U(z)$.

The minimum-norm method, as its name implies, seeks to minimize the norm of u in order to avoid spurious peaks in its pseudospectrum. Using (8.5.35), the norm of a vector u contained in the noise subspace is

$$\|u\|^2 = u^H u = u^H P_\omega u \tag{8.5.37}$$

However, an unconstrained minimization of this norm will produce the zero vector. Therefore, we place the constraint that the first element of u must equal 1.[5] This constraint can be expressed as

$$\delta_1^H u = 1 \tag{8.5.38}$$

where $\delta_1 = [1\ 0\ \cdots\ 0]^T$. Then the determination of the minimum-norm vector comes down to solving the following constrained minimization problem:

$$\min \|u\|^2 = u^H P_\omega u \qquad \text{subject to} \qquad \delta_1^H u = 1 \tag{8.5.39}$$

The solution can be found by using Lagrange multipliers (see Appendix B) and is given by

$$u_{mn} = \frac{P_\omega \delta_1}{\delta_1^H P_\omega \delta_1} \tag{8.5.40}$$

The frequency estimates are then obtained from the peaks in the pseudospectrum of the minimum-norm (mn) vector, u_{mn}

$$\bar{R}_{mn}(e^{j2\pi f}) = \frac{1}{|v^H(f)u_{mn}|^2} \tag{8.5.41}$$

The performance of the minimum-norm frequency estimation method is similar to that of MUSIC. For a performance comparison see Kaveh and Barabell (1986). Note that it is also possible to implement the

---

5 The choice of a value of 1 is somewhat arbitrary, since any nonzero constant will result in a similar solution.

minimum-norm method by rooting a polynomial rather than computing a psuedospectrum (see Problem 8.25).

EXAMPLE 8.5.3. In this example, we illustrate the use of the minimum-norm method and compare its performance to that of the other three frequency estimation methods discussed in this chapter: Pisarenko harmonic decomposition, the MUSIC algorithm, and the eigenvector method. The pseudospectrum of the minimum-norm method is found by first computing the minimum-norm vector $\mathbf{u}_{\text{mm}}$ and then finding its pseudospectrum, that is,

```
deltal=zeros(M, 1); deltal(1)=1;
Pn=Q(:,(P+1):M)*Q(:,(P+1):M)';      % noise subspace projection matrix
u=(Pn*el)/(el'*Pn*el);    % minimum-norm vector
Rbar=1./abs(fftshift(fft(u, Nfft))).^2;     % pseudospectrum
```

Consider the case of $P=4$ complex exponentials in noise with frequencies $f=0.1$, 0.25, 0.4, and -0.1, all with an amplitude of $\alpha=1$. The power of the noise is set to $\alpha_\omega^2=1$ with 100 realizations. The time-window length used was $M=8$ for all the methods except Pisarenko harmonic decomposition, which is constrained to use $M=P+1=5$. The pseudospectra are shown in Figure 8.21 with an FFT size of 1024, where we have not averaged in order to demonstrate the variance of the various methods. Here we see the large variance in the frequency estimates that is produced by Pisarenko harmonic decomposition compared to the other methods, which is a direct result of using a one-dimensional noise subspace. The other methods all perform comparably in terms of estimating the frequencies of the complex exponentials. Note the fluctuations in the pseudospectrum of the eigenvector method that result from the normalization by the eigenvalues. Since these eigenvalues vary over realizations, the pseudospectra will also reflect a similar variation. Routines for the eigenvector method and the minimum-norm method are provided in `ev_method.m` and `minnorm.m`, respectively.

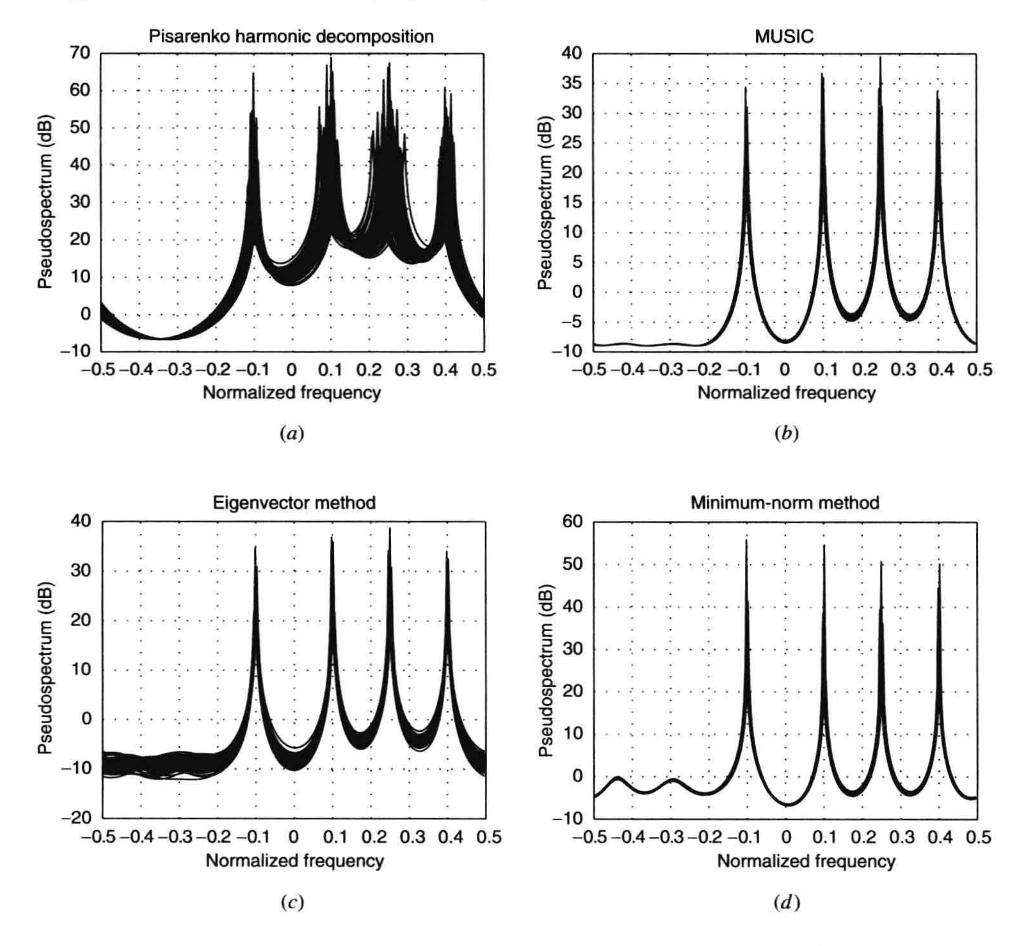


FIGURE 8.21

Comparison of the eigendecomposition-based frequency estimation methods: (*a*) Pisarenko harmonic decomposition, (*b*) MUSIC, (*c*) eigenvector method, and (*d*) minimum-norm method.

8.5.5 ESPRIT Algorithm

A frequency estimation technique that is built upon the same principles as other subspace methods but further exploits a deterministic relationship between subspaces is the *estimation of signal parameters via rotational invariance techniques (ESPRIT)* algorithm. This method differs from the other subspace methods discussed so far in this chapter in that the signal subspace is estimated from the data matrix $\boldsymbol{X}$ rather than the estimated correlation matrix $\hat{\boldsymbol{R}}_x$. The essence of ESPRIT lies in the rotational property between staggered subspaces that is invoked to produce the frequency estimates. In the case of a discrete-time signal or time series, this property relies on observations of the signal over two identical intervals staggered in time. This condition arises naturally for discrete-time signals, provided that the sampling is performed uniformly in time.[6] We first describe the original, least-squares version of the algorithm (Roy et al. 1986) and then extend the derivation to total least-squares ESPRIT (Roy and Kailath 1989), which is the preferred method for use. Since the derivation of the algorithm requires an extensive amount of formulation and matrix manipulations, we have included a block diagram in Figure 8.22 to be used as a guide through this process.

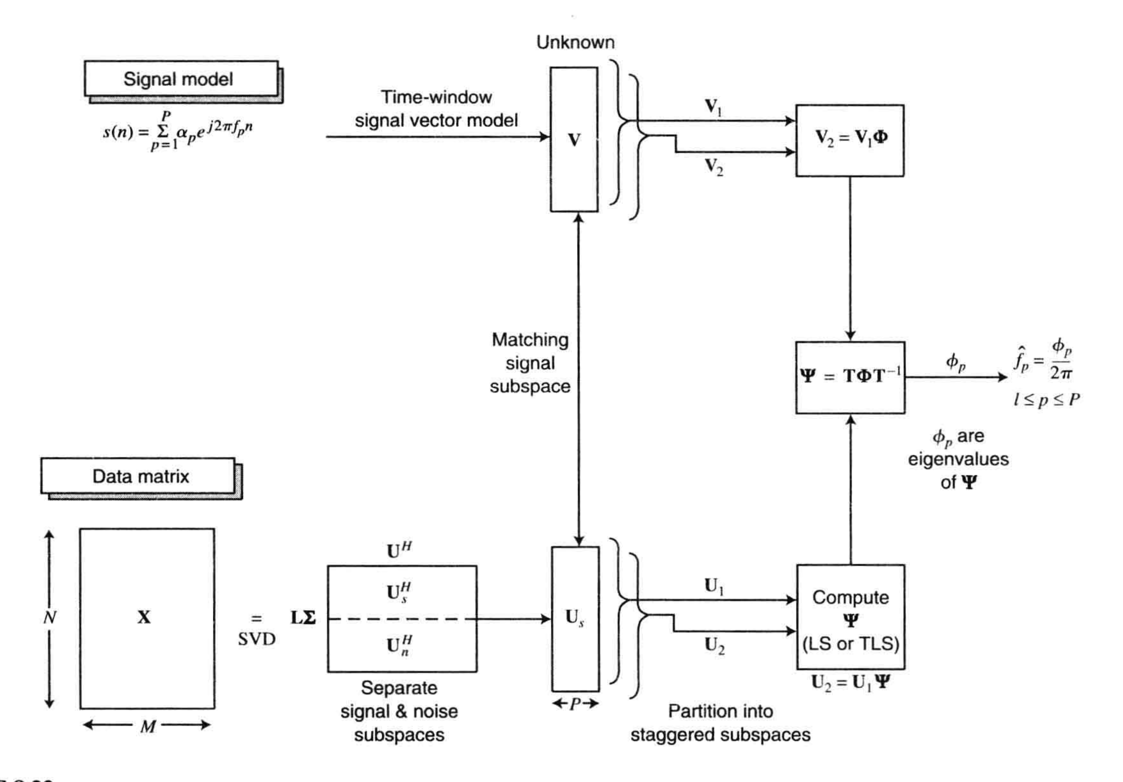


FIGURE 8.22
Block diagram demonstrating the flow of the ESPRIT algorithm starting from the data matrix through the frequency estimates.

Consider a single complex exponential $s_0(n) = \mathrm{e}^{\mathrm{j}2\pi fn}$ with complex amplitude α and frequency f. This signal has the following property

$$s_0(n+1) = \alpha \mathrm{e}^{\mathrm{j}2\pi f(n+1)} = s_0(n)\mathrm{e}^{\mathrm{j}2\pi f} \tag{8.5.42}$$

that is, the next sample value is a phase-shifted version of the current value. This phase shift can be represented as a rotation on the unit circle $\mathrm{e}^{\mathrm{j}2\pi f}$. Recall the time-window vector model from (8.5.4) consisting of a signal $\mathbf{s}(n)$, made up of complex exponentials, and the noise component $\boldsymbol{w}(n)$

[6]This condition is violated in the case of a nonuniformly sampled time series.

$$x(n) = \sum_{p=1}^{P} \alpha_p v(f_p) e^{j2\pi n f_p} + w(n) = V\Phi^n \alpha + w(n) = s(n) + w(n) \tag{8.5.43}$$

where the P columns of matrix V are length-M time-window frequency vectors of the complex exponentials

$$V = [v(f_1)\ v(f_2)\ \cdots\ v(f_P)] \tag{8.5.44}$$

The vector α consists of the amplitudes of the complex exponentials α_p. On the other hand, matrix Φ is the diagonal matrix of phase shifts between neighboring time samples of the individual, complex exponential components of $s(n)$

$$\Phi = \text{diag}\{\phi_1,\ \phi_2,\ \cdots,\ \phi_P\} = \begin{bmatrix} e^{j2\pi f_1} & 0 & \cdots & 0 \\ 0 & e^{j2\pi f_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & e^{j2\pi f_P} \end{bmatrix} \tag{8.5.45}$$

where $\phi_p = e^{j2\pi f_p}$ for $p = 1,\ 2,\ \cdots,\ P$. Since the frequencies of the complex exponentials f_p completely describe this rotation matrix, frequency estimates can be obtained by finding Φ. Let us consider two overlapping subwindows of length $M-1$ within the length M time-window vector. This subwindowing operation is illustrated in Figure 8.23. Consider the signal consisting of the sum of complex exponentials

$$s(n) = \begin{bmatrix} s_{M-1}(n) \\ s(n+M-1) \end{bmatrix} = \begin{bmatrix} s(n) \\ s_{M-1}(n+1) \end{bmatrix} \tag{8.5.46}$$

where $s_{M-1}(n)$ is the length-($M-1$) subwindow of $s(n)$, that is,

$$s_{M-1}(n) = V_{M-1}\Phi^n \alpha \tag{8.5.47}$$

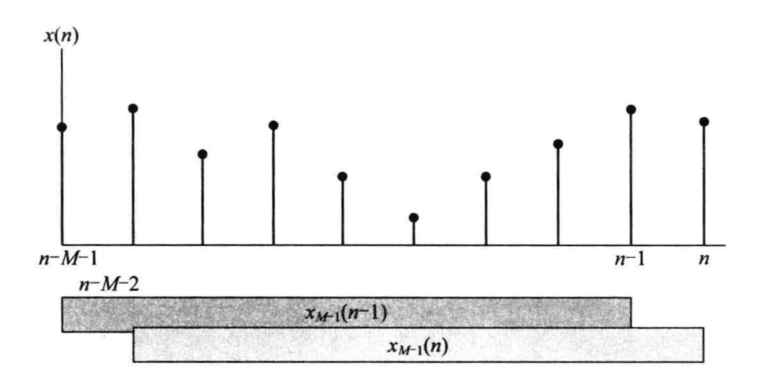


FIGURE 8.23
Time-staggered, overlapping windows used by the ESPRIT algorithm.

Matrix V_{M-1} is constructed in the same manner as **V** except its time-window frequency vectors are of length $M-1$, denoted as $v_{M-1}(f)$,

$$V_{M-1} = [v_{M-1}(f_1)\ v_{M-1}(f_2)\ \cdots\ v_{M-1}(f_P)] \tag{8.5.48}$$

Recall that $s(n)$ is the scalar signal made up of the sum of complex exponentials at time n. Using the relation in (8.5.47), we can define the matrices

$$V_1 = V_{M-1}\Phi^n \quad V_2 = V_{M-1}\Phi^{n+1} \tag{8.5.49}$$

where V_1 and V_2 correspond to the unstaggered and staggered windows, that is,

$$V = \begin{bmatrix} V_1 \\ ** \cdots * \end{bmatrix} = \begin{bmatrix} ** \cdots * \\ V_2 \end{bmatrix} \tag{8.5.50}$$

Clearly, by examining (8.5.49), these two matrices of time-window frequency vectors are related as

$$V_2 = V_1 \Phi \tag{8.5.51}$$

Note that each of these two matrices spans a different, though related, $(M-1)$-dimensional subspace.

Now suppose that we have a data matrix X from (8.5.20) with N data records of the length-M time-window vector signal $\mathbf{x}(n)$. Using the singular value decomposition (SVD) discussed in Chapter 7, we can write the data matrix as[7]

$$\mathbf{X} = \mathbf{L}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{H}} \tag{8.5.52}$$

where L is an $N \times N$ matrix of left singular vectors and $\mathbf{U}$ is an $M \times M$ matrix of right singular vectors. Both of these matrices are unitary; that is, $V^{\mathrm{H}}V = I$ and $U^{H}U = I$. The matrix Σ has dimensions $N \times M$ consisting of singular values on the main diagonal ordered in descending magnitude. The squared magnitudes of the singular values are equal to the eigenvalues of R scaled by a factor of N from (8.5.21), and the columns of U are their corresponding eigenvectors. Thus, U forms an orthonormal basis for the underlying M-dimensional vector space. This subspace can be partitioned into signal and noise subspaces as

$$U = [U_s \,|\, U_n] \tag{8.5.53}$$

where U_s is the matrix of right-hand singular vectors corresponding to the singular values with the P largest magnitudes. Note that since the signal portion consists of the sum of complex exponentials modeled as time-window frequency vectors $v(f)$, all these frequency vectors, for $f = f_1, f_2, \cdots, f_P$, must also lie in the signal subspace. As a result, the matrices V and U_s span the same subspace. Therefore, there exists an invertible transformation T that maps U_s into V, that is,

$$V = U_s T \tag{8.5.54}$$

The transformation T is never solved for in this derivation, but instead is only formulated as a mapping between these two matrices within the signal subspace.

Proceeding as we did with the matrix V in (8.5.50), we can partition the signal subspace into two smaller $(M-1)$-dimensional subspaces as

$$U_s = \begin{bmatrix} U_1 \\ ** \cdots * \end{bmatrix} = \begin{bmatrix} ** \cdots * \\ U_2 \end{bmatrix} \tag{8.5.55}$$

where U_1 and U_2 correspond to the unstaggered and staggered subspaces, respectively. Since V_1 and V_2 correspond to the same subspaces, the relation from (8.5.54) must also hold for these subspaces

$$V_1 = U_1 T \qquad V_2 = U_2 T \tag{8.5.56}$$

The staggered and unstaggered components of the matrix V in (8.5.50) are related through the subspace rotation Φ in (8.5.51). Since the matrices U_1 and U_2 also span these respective, related subspaces, a similar, though different, rotation must exist that relates (rotates) U_1 to U_2

$$U_2 = U_1 \Psi \tag{8.5.57}$$

where Ψ is this rotation matrix.

Recall that frequency estimation comes down to solving for the subspace rotation matrix Φ. We can estimate Φ by making use of the relations in (8.5.56) together with the rotations between the staggered signal subspaces in

---

[7]Our notation differs slightly from that introduced in Chapter 8 in order to avoid confusion with the matrix of time-window frequency vectors V.

(8.5.51) and (8.5.57). In this process, the matrices U_1 and U_2 are known from the SVD on data matrix X. First, we solve for Ψ from the relation in (8.5.57), using the method of least-squares (LS) from Chapter 7

$$\Psi = (U_1^H U_1)^{-1} U_1^H U_2 \quad (8.5.58)$$

Substituting (8.5.57) into (8.5.56), we have

$$V_2 = U_2 T = U_1 \Psi T \quad (8.5.59)$$

Similarly, we can also solve for V_2 using the relation in (8.5.51) and substituting (8.5.56) for V_1

$$V_2 = V_1 \Phi = U_1 T \Phi \quad (8.5.60)$$

Thus, equating the two right-hand sides of (8.5.59) and (8.5.60), we have the following relation between the two subspace rotations

$$\Psi T = T \Phi \quad (8.5.61)$$

or equivalently

$$\Psi = T \Phi T^{-1} \quad (8.5.62)$$

Equations (8.5.61) and (8.5.62) should be recognized as the relationship between eigenvectors and eigenvalues of the matrix Ψ (Golub and Van Loan 1996). Therefore, the diagonal elements of Φ, ϕ_p for $p = 1, 2, \cdots, P$, are simply the eigenvalues of Ψ. As a result, the estimates of the frequencies are

$$\hat{f}_p = \frac{\measuredangle \phi_p}{2\pi} \quad (8.5.63)$$

where $\measuredangle \phi_p$ is the phase of ϕ_p. Although the principle behind the ESPRIT algorithm, namely, the use of subspace rotations, is quite simple, one can easily get lost in the details of the derivation of the algorithm. Note that we have only used simple matrix relationships. An illustrative example of the implementation of ESPRIT in MATLAB is given in Example 8.6.4 to help clarify the details of the algorithm. However, first we give a total least-squares version of the algorithm, which is the preferred method for use.

Note that the subspaces U_1 and U_2 are *both* only estimates of the true subspaces that correspond to V_1 and V_2 respectively, obtained from the data matrix X. The estimate of the subspace rotation was obtained by solving (8.5.57) using the LS criterion

$$\Psi_{ls} = (U_1^H U_1)^{-1} U_1^H U_2 \quad (8.5.64)$$

This LS solution is obtained by minimizing the errors in an LS sense from the following formulation

$$U_2 + E_2 = U_1 \Psi \quad (8.5.65)$$

where $\mathbf{E}_2$ is a matrix consisting of errors between $\mathbf{U}_2$ and the true subspace corresponding to V_2. Note that this LS formulation assumes errors only on the estimation of $\mathbf{U}_2$ and no errors between $\mathbf{U}_1$ and the true subspace that it is attempting to estimate corresponding to $\mathbf{V}_1$. Therefore, since $\mathbf{U}_1$ is also an estimated subspace, a more appropriate formulation is

$$U_2 + E_2 = (U_1 + E_1)\Psi \quad (8.5.66)$$

where E_1 is the matrix representing the errors between U_1 and the true subspace corresponding to V_1. A solution to this problem, known as *total least squares (TLS)*, is obtained by minimizing the Frobenius norm of the two error matrices

$$\| E_1 \quad E_2 \|_F \quad (8.5.67)$$

Since the principles of TLS are beyond the scope of this book, we simply give the procedure to obtain the TLS solution of Ψ and refer the interested reader to Golub and Van Loan (1996).

First, form a matrix made up of the staggered signal subspace matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ placed side by side,[8] and perform an SVD

[8]Note that this matrix $[\mathbf{U}_1 \mathbf{U}_2] \neq \mathbf{U}_s = [\mathbf{U}_1^T \mathbf{U}_2^T]^T$ from (8.6.55).

$$[\boldsymbol{U}_1 \boldsymbol{U}_2] = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{U}}^{\mathrm{H}} \tag{8.5.68}$$

We then operate on the $2P \times 2P$ matrix $\tilde{\boldsymbol{U}}$ of right singular vectors. This matrix is partitioned into $P \times P$ quadrants

$$\tilde{\boldsymbol{U}} = \begin{bmatrix} \tilde{\boldsymbol{U}}_{11} & \tilde{\boldsymbol{U}}_{12} \\ \tilde{\boldsymbol{U}}_{21} & \tilde{\boldsymbol{U}}_{22} \end{bmatrix} \tag{8.5.69}$$

The TLS solution for the subspace rotation matrix $\boldsymbol{\Psi}$ is then

$$\boldsymbol{\Psi}_{\text{tls}} = -\tilde{\boldsymbol{U}}_{12}\tilde{\boldsymbol{U}}_{22}^{-1} \tag{8.5.70}$$

The frequency estimates are then obtained from (8.5.62) and (8.5.63) by using Ψ_{tls} from (8.5.70). Although the TLS version of ESPRIT involves slightly more computations, it is generally preferred over the LS version based on formulation in (8.5.66). A statistical analysis of the performance of the ESPRIT algorithms is given in Ottersten et al. (1991).

EXAMPLE 8.5.4. In this illustrative example, we demonstrate the use of both the LS and TLS versions of the ESPRIT algorithm on a set of complex exponentials in white noise using MATLAB. First, generate a signal $s(n)$ of length $N = 128$ consisting of complex exponential signals at normalized frequencies f = 0.1, 0.15, 0.4, and −0.15, all with amplitude $\alpha = 1$. Each of the complex exponentials is generated by `exp(j*2*pi*f*[0:(N-1)]');`. The overall signal in white noise with unit power ($\sigma_w^2 = 1$) is then

```
x=s+(randn(N,1)+j*randn(N,1))/sqrt(2)
```

We form the data matrix corresponding to (8.5.20) for a time window of length M = 8. The least-squares ESPRIT algorithm is then performed as follows:

```
[L, S, U]=svd(X);
Us=U(:, 1:P);    % signal subspace
U1=Us(1:M-1),:);    U2=Us(2:M,:);    % signal subspaces
Psi=U1\U2;    % TLS solution for psi
```

If we are using the TLS version of ESPRIT, then solve for

```
[LL, SS, UU]=svd([U1  U2]); UU12=UU(1:P, (P+1):(2*P));
UU22=UU((p+1):(2*p), (p+1):(2*p));
Psi=-UU12*inv(UU22);    % TLS solution for Psi
```

The frequencies are found by computing the phases of the eigenvalues of $\boldsymbol{\Psi}$, that is,

```
phi=eig(Psi);   % eigenvalues of Psi
fhat=angle(phi)/(2*pi);    % frequency estimates
```

In both cases, we average over 1000 realizations and obtain average estimated frequencies very close to the true values f = 0.1, 0.15, 0.4, and −0.15 used to generate the signals. Routines for both the LS and TLS versions of ESPRIT are provided in `esprit_ls.m` and `esprit_tls.m`.

8.6 Summary

In this chapter, we have examined the modeling process for both pole-zero and harmonic signal models. As for all signal modeling problems, the procedure begins with the selection of the appropriate model for the signal under consideration. Then the signal model is applied by estimating the model parameters from a collection of data samples. However, as we have stressed throughout this chapter, nothing is more valuable in the modeling process than specific knowledge of the signal and its underlying process in order to assess the validity of the model for a particular signal. For this reason, we began the chapter with a discussion of a model building procedure, starting with the choice of the appropriate model and the estimation of its parameters, and concluding with the validation of the model. Clearly, if the model is not well-suited for the signal, the application of the model becomes meaningless.

In the first part of the chapter, we considered the application of the parametric signal models that were discussed in Chapter 3. The estimation of all-pole models was presented for both direct and lattice structures. Within this context, we used various model order selection criteria to determine the order of the all-pole model. However, these criteria are not necessarily limited to all-pole models. In addition, the relationship was given between the all-pole model and Burg's method of maximum entropy. Next, we considered the pole-zero modeling. Using a nonlinear least-squares technique, a method was presented for estimating the parameters of the pole-zero model. The use of

pole-zero models for the purposes of spectral estimation along with their application to speech modeling was also considered.

The latter part of the chapter focused on harmonic signal models, that is, modeling signals using the sum of complex exponentials. The harmonic modeling problem becomes one of estimating the frequency of the complex exponentials. Then, we discuss some of the more popular harmonic modeling methods. Starting with the Pisarenko harmonic decomposition, the first such model, we discuss the MUSIC, eigenvector, root-MUSIC, and minimum-norm methods for frequency estimation. All of these methods are based on computing a pseudospectrum or a rooting polynomial from an estimated correlation matrix. Finally, we give a brief derivation of the ESPRIT algorithm, both in its original LS form and the more commonly used TLS form.

Problems

8.1 Consider the random process $x(n)$ described in Example 8.2.3 that is simulated by exciting the system function

$$H(z)=\frac{1}{1-2.7607z^{-1}+3.8108z^{-2}-2.6535z^{-3}+0.9238z^{-4}}$$

using a WGN(0, 1) process. Generate $N = 250$ samples of the process $x(n)$.

(*a*) Write a MATLAB function that implements the modified covariance method to obtain AR(*P*) model coefficients and the modeling error variance $\hat{\sigma}_P^2$ as a function of *P*, using *N* samples of $x(n)$.

(*b*) Compute and plot the variance $\hat{\sigma}_P^2$, FPE(*P*), AIC(*P*), MDL(*P*), and CAT(*P*) for $P=1,2,\cdots,15$.

(*c*) Comment on your results and the usefulness of model selection criteria for the process $x(n)$.

8.2 Consider the Burg approach of minimizing forward-backward LS error $\varepsilon_m^{\text{fb}}$ in (8.2.33).

(*a*) Show that by using (8.2.26) and (8.2.27), $\varepsilon_m^{\text{fb}}$ can be put in the form of (8.2.34).

(b) By minimizing $\varepsilon_m^{\text{fb}}$ with respect to k_{m-1}, show that the expression for the optimum k_{m-1}^B is given by (8.2.35).

(c) Show that $|k_{m-1}^B|<1$.

(d) Show that $|k_{m-1}^B|<|k_{m-1}^{IS}|\le 1$ where k_{m-1}^{IS} is defined in (8.2.36).

8.3 Generate an AR(2) process using the system function

$$H(z)=\frac{1}{1-0.9z^{-1}+0.81z^{-2}}$$

excited by a WGN(0, 1) process. Illustrate numerically that if we use the full-windowing method, that is, the matrix $\bar{\mathbf{X}}$ in (8.2.8), then the PACS estimates $\{k_m^{\text{FM}}\}_{m=0}^{1}$, $\{k_m^{\text{BM}}\}_{m=0}^{1}$, and $\{k_m^{\text{B}}\}_{m=0}^{1}$ of Section 8.2 are identical and hence can be obtained by using the Levinson-Durbin algorithm.

8.4 Generate sample sequences of an AR(2) process

$$x(n)=\omega(n)-1.5857x(n-1)-0.9604x(n-2)$$

where $\omega(n)\sim \text{WGN}(0,1)$. Choose $N=256$ samples for each realization.

(a) Design a first-order optimum linear predictor, and compute the prediction error $e_1(n)$. Test the whiteness of the error sequence $e_1(n)$ using the autocorrelation, PSD, and partial correlation methods, discussed in Section 8.1. Show your results as an overlay plot using 20 realizations.

(b) Repeat the above part, using second- and third-order linear predictors.

(c) Comment on your plots.

8.5 Generate sample functions of the process

$$x(n)=0.5\omega(n)+0.5\omega(n-1)$$

where $\omega(n)\sim \text{WGN}(0,1)$. Choose $N=256$ samples for each realization.

(a) Test the whiteness of $x(n)$ and show your results, using overlay plots based on 10 realizations.

(b) Process $x(n)$ through the AR(1) filter

$$H(z)=\frac{1}{1+0.95z^{-1}}$$

to obtain $y(n)$. Test the whiteness of $y(n)$ and show your results, using overlay plots based on 10 realizations.

8.6 The process $x(n)$ contains a complex exponential in white noise, that is,

$$x(n) = A\mathrm{e}^{\mathrm{j}(\omega_0 n+\theta)} + \omega(n)$$

where A is a real positive constant, θ is a random variable uniformly distributed over $[0, 2\pi]$, ω_0 is a constant between 0 and π, and $\omega(n) \sim \mathrm{WGN}(0, \sigma_\omega^2)$. The purpose of this problem is to analytically obtain a maximum entropy method (MEM) estimate by fitting an AR(P) model and then evaluating $\{a_k\}_0^P$ model coefficients.

(a) Show that the $(P+1)\times(P+1)$ autocorrelation matrix of $x(n)$ is given by

$$\boldsymbol{R}_x = A^2 \boldsymbol{e}\boldsymbol{e}^{\mathrm{H}} + \sigma_\omega^2 \boldsymbol{I}$$

where $\mathbf{e} = [1\ \mathrm{e}^{-\mathrm{j}\omega_0}\ \cdots\ \mathrm{e}^{-\mathrm{j}P\omega_0}]^{\mathrm{T}}$.

(b) By solving autocorrelation normal equations, show that

$$\begin{aligned}\mathbf{a}_P &\triangleq [1\ a_1\ \cdots\ a_P]^{\mathrm{T}} \\ &= \left(1 + \frac{A^2}{\sigma_\omega^2 + A^2 P}\right)\left[\mathbf{e} - \frac{A^2}{\sigma_\omega^2 + (P+1)A^2}[1\ 0\ \cdots\ 0]^{\mathrm{T}}\right]\end{aligned}$$

(c) Show that the MEM estimate based on the above coefficients is given by

$$\hat{R}_x(\mathrm{e}^{\mathrm{j}\omega}) = \frac{\sigma_\omega^2\left[1 - \frac{A^2}{\sigma_\omega^2+(P+1)A^2}\right]}{|1 - \frac{A^2}{\sigma_\omega^2+(P+1)A^2} W_R(\mathrm{e}^{\mathrm{j}(\omega-\omega_0)})|^2}$$

where $W_R(\mathrm{e}^{\mathrm{j}\omega})$ is the DTFT of the $(P+1)$ length rectangular window.

8.7 An AR(2) process $y(n)$ is observed in noise $v(n)$ to obtain $x(n)$, that is,

$$x(n) = y(n) + v(n) \qquad v(n) \sim WGN(0, \sigma_v^2)$$

where $v(n)$ is uncorrelated with $y(n)$ and

$$y(n) = 1.27y(n-1) - 0.81y(n-2) + \omega(n) \qquad \omega(n) \sim \mathrm{WGN}(0,1)$$

(a) Determine and plot the true power spectrum $R_x(\mathrm{e}^{\mathrm{j}\omega})$.

(b) Generate 10 realizations of $x(n)$, each with $N = 256$ samples. Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 2$ and $\sigma_v^2 = 1$. Obtain an overlay plot of this estimate, and compare it with the true spectrum.

(c) Repeat part (*b*), using $\sigma_v^2 = 10$. Comment on the effect of increasing noise variance on spectrum estimates.

(d) Since the noise variance σ_v^2 affects only $r_x(0)$, investigate the effect of subtracting a small amount from $r_x(0)$ on the spectrum estimates in part (*c*).

8.8 Let $x(n)$ be a random process whose correlation is estimated. The values for the first five lags are $r_x(0) = 1$, $r_x(1) = 0.7$, $r_x(2) = 0.5$, $r_x(3) = 0.3$, and $r_x(4) = 0$.

(a) Determine and plot the Blackman-Tukey power spectrum estimate.

(b) Assume that $x(n)$ is modeled by an AP(2) model. Determine and plot its spectrum estimate.

(c) Now repeat (b) assuming that AP(4) is an appropriate model for $x(n)$. Determine and plot the spectrum estimate.

8.9 The narrowband process $x(n)$ is generated using the AP(4) model

$$H(z) = \frac{1}{1 + 0.98z^{-1} + 1.92z^{-2} + 0.94z^{-3} + 0.92z^{-4}}$$

driven by WGN(0, 0.001). Generate 10 realizations, each with $N = 256$ samples, of this process.

(a) Determine and plot the true power spectrum $R_x(\mathrm{e}^{\mathrm{j}\omega})$.

(b) Using the LS approach with forward linear predictor, estimate the power spectrum for P = 4. Obtain an overlay plot of this estimate, and compare it with the true spectrum.

(c) Repeat part (*b*) with P = 8 and 12. Provide a qualitative description of your results with respect to model order size.

(d) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for P = 4. Obtain an overlay plot of this estimate. Compare it with the plot in part (*b*).

8.10 Consider the following PZ(4, 2) model

$$H(z) = \frac{1-z^{-2}}{1+0.41z^{-4}}$$

driven by $\text{WGN}(0,1)$ to obtain a broadband ARMA process $x(n)$. Generate 10 realizations, each with $N = 256$ samples, of this process.

(a) Determine and plot the true power spectrum $R_x(e^{j\omega})$.

(b) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 12$. Obtain an overlay plot of this estimate, and compare it with the true spectrum.

8.11 A random process $x(n)$ is given by

$$x(n) = \cos(\frac{\pi n}{3} + \theta_1) + \omega(n) - \omega(n-2) + \cos(\frac{2\pi n}{3} + \theta_2)$$

where $\omega(n) \sim \text{WGN}(0,1)$ and θ_1 and θ_2 are IID random variables uniformly distributed between 0 and 2π. Generate a sample sequence with $N = 256$ samples.

(a) Determine and plot the true spectrum $R_x(e^{j\omega})$.

(b) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 10$, 20, and 40 from the generated sample sequence. Compare it with the true spectrum.

8.12 Show that, for large values of N, the modeling error variance estimate given by Equation (8.2.38) can be approximated by the estimate given by Equation (8.2.39).

8.13 This problem investigates the effect of correlation aliasing observed in LS estimation of model parameters when the AP model is excited by discrete spectra. Consider an AP(1) model with pole at $z = \alpha$ excited by a periodic sequence of period N. Let $x(n)$ be the output sequence.

(a) Show that the correlation at lag 1 satisfies

$$r_x(1) = \frac{\alpha^{N-1} + \alpha}{1+\alpha^N} r_x(0) \tag{P.1}$$

(b) Using the LS approach, determine the estimate $\hat{\alpha}$ as a function of α and N. Compute $\hat{\alpha}$ for $\alpha = 0.9$ and $N = 10$.

(c) Generate $x(n)$, using $\alpha = 0.95$ and the periodic impulse train with $N = 10$. Compute and plot the correlation sequence $r_x(l)$, $0 \le l \le N-1$, of $x(n)$,. Compare your plot with the AP(1) model correlation for $\alpha = 0.95$ Comment on your observations and discuss why they explain the discrepancy between α and $\hat{\alpha}$.

(d) Repeat part (c) for $N = 100$ and 1000. Show analytically and numerically that $\hat{\alpha} \to \alpha$ as $N \to \infty$.

8.14 In this problem, we investigate the equation error method of Section 8.3.1. Consider the PZ(2, 2) model

$$x(n) = 0.3x(n-1) + 0.4x(n-2) + \omega(n) + 0.25\omega(n-2)$$

Generate $N = 200$ samples of $x(n)$, using $\omega(n) \sim \text{WGN}(0, \sqrt{10})$. Record values of both $x(n)$ and $\omega(n)$.

(a) Using the residual windowing method, that is, $N_i = \max(P, Q)$ and $N_f = N-1$, compute the estimates of the above model parameters.

(b) Compute the input variance estimate $\hat{\sigma}_\omega^2$ from your estimated values in part (a). Compare it with the actual value σ_ω^2 and with (8.3.12).

8.15 Consider the following PZ(4, 2) model

$$x(n) = 1.8766x(n-1) - 2.6192x(n-2) + 1.6936x(n-3) - 0.8145x(n-4)$$
$$+\omega(n) + 0.05\omega(n-1) - 0.855\omega(n-2)$$

excited by $\omega(n) \sim \text{WGN}(0, \sqrt{10})$. Generate 300 samples of $x(n)$.

(a) Assuming the AP(10) model for the data segment, estimate its parameters by using the LS approach described in Section 8.2.

(b) Generate a plot similar to Figure 8.13 by computing spectra corresponding to the true PZ(4, 2), estimated PZ(4, 2), and estimated AP(10) models. Compare and comment on your results.

8.16 Using matrix notation, show that AZ power spectrum estimation is equivalent to the Blackman-Tukey method discussed in Chapter 4.

8.17 Consider the PZ(4, 2) model given in Problem 8.15. Generate 300 samples of $x(n)$.
(a) Fit an AP(5) model to the data and plot the resulting spectrum.
(b) Fit an AP(10) model to the data and plot the resulting spectrum.
(c) Fit an AP(50) model to the data and plot the resulting spectrum.
(d) Compare your plots with the true spectrum, and discuss the effect of model mismatch on the quality of the spectrum.

8.18 Use the supplied (about 50-ms) segment of a speech signal sampled at 8192 samples per second.
(a) Compute a periodogram of the speech signal (see Chapter 4).
(b) Using data windowing, fit an AP(16) model to the speech data and compute the spectrum.
(c) Using the residual windowing, fit a PZ(12, 6) model to the speech data and compute the spectrum.
(d) Plot the above three spectra on one graph, and comment on the performance of each method.

8.19 One practical approach to spectrum estimation discussed in Section 8.4 is the prewhitening and postcoloring method.
(a) Develop a MATLAB function to implement this method. Use the forward/backward LS method to determine AP(P) parameters and the Welch method for nonparametric spectrum estimation.
(b) Verify your function on the short segment of the speech segment from Problem 8.18.
(c) Compare your results with those obtained in Problem 8.18.

8.20 Consider a white noise process with variance σ_ω^2. Find its minimum-variance power spectral estimate.

8.21 Find the minimum-variance spectrum of a first-order all pole model, that is,

$$x(n) = -a_1 x(n-1) + \omega(n)$$

8.22 The filter coefficient vector for the minimum-variance spectrum estimator is given in (8.5.10). Using Lagrange multipliers, discussed in Appendix B, solve this constrained optimization to find this weight vector.

8.23 Using the relationship between the minimum-variance and the all-pole model spectrum estimators in (8.5.22), generate a recursive relationship for the minimum-variance spectrum estimators of increasing window length. In other words, write $\hat{R}_{M+1}^{(\mathrm{mv})}(e^{j2\pi f})$ in terms of $\hat{R}_M^{(\mathrm{mv})}(e^{j2\pi f})$ and the all-pole model spectrum estimator $\hat{R}_M^{(\mathrm{ap})}(e^{j2\pi f})$ in (8.5.20).

8.24 In Pisarenko harmonic decomposition, discussed in Section 8.6.2, we determine the frequencies of the complex exponentials in white noise through the use of the pseudospectrum. The word *pseudospectrum* was used because its value does not correspond to an estimated power. Find a set of linear equations that can be solved to find the powers of the complex exponentials. *Hint:* Use the relationship of eigenvalues and eigenvectors $\boldsymbol{R}_x \boldsymbol{q}_m = \lambda_m \boldsymbol{q}_m$ for $m = 1, 2, \cdots, M$.

8.25 For the MUSIC algorithm, we showed a means of using the MUSIC pseudospectrum to derive a polynomial that could be rooted to obtain frequency estimates, which is known as root-MUSIC. Find a similar rooting method for the minimum-norm frequency estimation procedure.

8.26 The Pisarenko harmonic decomposition, MUSIC, and minimum-norm algorithms yield frequency estimates by computing a pseudospectrum using the Fourier transforms of the eigenvectors. However, these pseudospectra do not actually estimate a power. Derive the minimum-variance spectral estimator in terms of the Fourier transforms of the eigenvectors and the associated eigenvalues. Relate this result to the MUSIC and eigenvector method pseudospectra.

8.27 Show that the pseudospectrum for the MUSIC algorithm is equivalent to the minimum-variance spectrum in the case of an infinite signal-to-noise ratio.

8.28 Find a relationship between the minimum-norm pseudospectrum and the MUSIC pseudospectrum. What are the implications of this relationship?.

8.29 In (8.5.22), we derived a relationship between the minimum-variance spectral estimator and spectrum estimators derived from all-pole models of orders 1 to M. Find a similar relationship between the pseudospectra of the MUSIC and minimum-norm algorithms that shows that the MUSIC pseudospectrum is a weighted average of minimum-norm pseudospectra.

CHAPTER 9

Adaptive Filters

In Chapter 1, we discussed different practical applications that demonstrated the need for adaptive filters, pointed out the key aspects of the underlying *signal operating environment* (*SOE*), and illustrated the key features and types of adaptive filters. The defining characteristic of an adaptive filter is its ability to operate satisfactorily, according to a criterion of performance acceptable to the user, in an unknown and possibly time-varying environment without the intervention of the designer. In Chapter 5, we developed the theory of optimum filters under the assumption that the filter designer has complete knowledge of the statistical properties (usually second-order moments) of the SOE. However, in real-world applications such information is seldom available, and the most practical solution is to use an adaptive filter. Adaptive filters can improve their performance, during normal operation, by learning the statistical characteristics through processing current signal observations.

In this chapter, we develop a mathematical framework for the design and performance evaluation of adaptive filters, both theoretically and by simulation. The goal of an adaptive filter is to "find and track" the optimum filter corresponding to the same signal operating environment with complete knowledge of the required statistics. In this context, optimum filters provide both guidance for the development of adaptive algorithms and a yardstick for evaluating the theoretical performance of adaptive filters. We start in Section 9.1 with discussion of a few typical application problems that can be effectively solved by using an adaptive filter. The performance of adaptive filters is evaluated using the concepts of stability, speed of adaptation, quality of adaptation, and tracking capabilities. These issues and the key features of an adaptive filter are discussed in Section 9.2. Since most adaptive algorithms originate from deterministic optimization methods, in Section 9.3 we introduce the family of steepest-descent algorithms and study their properties. Sections 9.4 and 9.5 provide a detailed discussion of the derivation, properties, and applications of the two most important adaptive filtering algorithms: the least mean square (LMS) and the recursive least-squares (RLS) algorithms. Section 9.6 provides fast implementations of the RLS algorithm for the FIR filtering case. The development of the later algorithms is a result of the shift invariance of the data stored in the memory of the FIR filter. Finally, in Section 9.7 we provide a concise introduction to the tracking properties of the LMS and the RLS algorithms.

9.1 Typical Applications of Adaptive Filters

As we have already seen in Chapter 1, many practical applications cannot be successfully solved by using fixed digital filters because either we do not have sufficient information to design a digital filter with fixed coefficients or the design criteria change during the normal operation of the filter. Most of these applications can be successfully solved by using a special type of "smart" filters known collectively as *adaptive filters*. The distinguishing feature of adaptive filters is that they can modify their response to improve their performance during operation without any intervention from the user.

The best way to introduce adaptive filters is with some applications for which they are well suited. These and other applications are discussed in greater detail in the sequel as we develop the necessary background and tools.

9.1.1 Echo Cancelation in Communications

An echo is the delayed and distorted version of an original signal that returns to its source. In some applications (radar, sonar, or ultrasound), the echo is the wanted signal; however, in communication applications, the echo is an unwanted signal that must be eliminated. There are two types of echoes in communication systems: (1) *electrical* or *line echoes*, which are generated electrically due to impedance mismatches at points along the transmission medium, and (2) *acoustic echoes*, which result from the reflection of sound waves and acoustic coupling between a microphone and a loudspeaker.

Here we focus on electrical echoes in voice communications; electrical echoes in data communications are discussed in Section 9.4.4, and acoustic echoes in teleconferencing and hands-free telephony were discussed in Section 1.4.1.

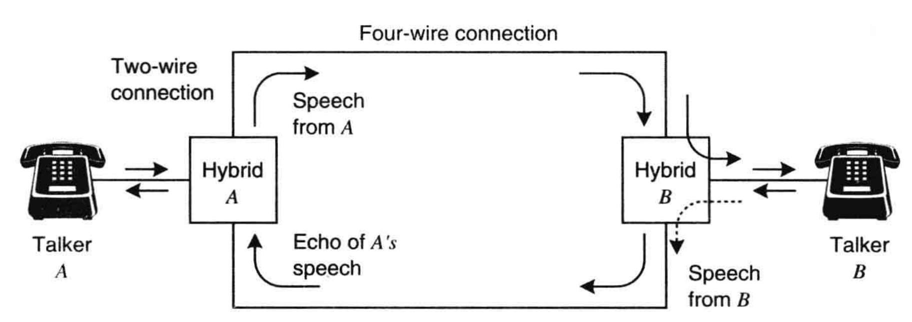


FIGURE 9.1
Echo generation in a long-distance telephone network.

Electrical echoes are observed on long-distance telephone circuits. A simplified form of such a circuit, which is sufficient for the present discussion, is shown in Figure 9.1. The local links from the customer to the telephone office consist of bidirectional two-wire connections, whereas the connection between the telephone offices is a four-wire carrier facility that may include a satellite link. The conversion between two-wire and four-wire links is done by special devices known as *hybrids*. An ideal hybrid should pass (1) the incoming signal to the two-wire output without any leakage into its output port and (2) the signal from the two-wire circuit to its output port without reflecting any energy back to the two-wire line (Sondhi and Berkley 1980). In practice, due to impedance mismatches, the hybrids do not operate perfectly. As a result, some energy on the incoming branch of the four-wire circuit leaks into the outgoing branch and returns to the source as an echo (see Figure 9.1). This echo, which is usually 11 dB down from the original signal, makes it difficult to carry on a conversation if the round-trip delay is larger than 40 ms. Satellite links, as a consequence of high altitude, involve round-trip delays of 500 to 600 ms.

The first devices used by telephone companies to control voice echoes were echo suppressors. Basically, an *echo suppressor* is a voice-activated switch that attempts to impose an open circuit on the return path from listener to talker when the listener is silent (see Figure 9.2). The main problems with these devices are speech clipping during double-talking and the inability to effectively deal with round-trip delays longer than 100 ms (Weinstein 1977).

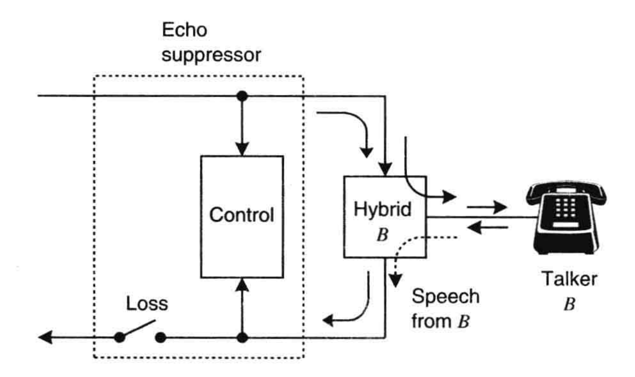


FIGURE 9.2
Principle of echo suppression.

The problems associated with echo suppressors could be largely avoided if we could estimate the *transmission path* from point C to point D (see Figure 9.3), which is known as the *echo path*. If we knew the echo path, we could design a filter that produced a copy or replica of the echo signal when driven by the signal at point C. Subtraction of the echo replica from the signal at point D will eliminate the echo without distorting the speech of the second talker that may be present at point D. The resulting device, shown in Figure 9.3, is known as an *echo canceler*.

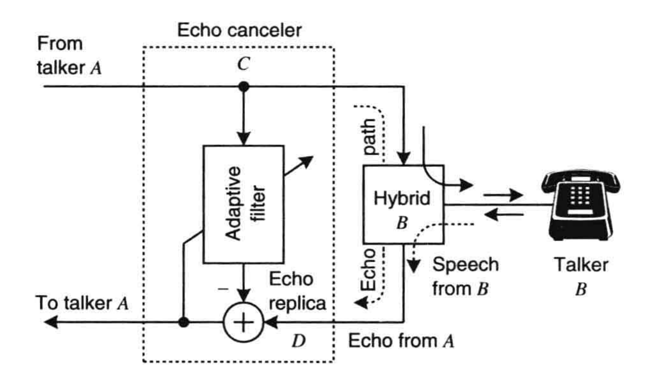


FIGURE 9.3
Principle of echo cancelation.

In practice, the channel characteristics are generally *not* known. For dial-up telephone lines, the channel differs from call to call, and the characteristics of radio and microwave channels (phase perturbations, fading, etc.) change significantly with time. Therefore, we *cannot* design and use a *fixed* echo canceler with satisfactory performance for all possible connections. There are two possible ways around this problem:

1. Design a compromise *fixed* echo canceler based on some "average" echo path, assuming that we have sufficient information about the connections to be seen by the canceler.
2. Design an *adaptive echo canceler* that can "learn" the echo path when it is first turned on and afterward "tracks" its variations without any intervention from the designer. Since an adaptive canceler matches the echo path for any given connection, it performs better than a compromise fixed canceler.

We stress that the main task of the canceler is to estimate the echo signal with sufficient accuracy; the estimation of the echo path is simply the means of achieving this goal. The performance of the canceler is measured by the attenuation, in decibels, of the echo, which is known as *echo return loss enhancement.* The adaptive echo canceler achieves this goal by modifying its response, using the residual echo signal in an as yet unspecified way.

Adaptive echo cancelers are widely used in voice telecommunications, and the international standards organization CCITT has issued a set of recommendations (CCITT G. 165) that outlines the basic requirements for echo cancelers. More details can be found in Weinstein (1977) and Murano et al. (1990).

9.1.2 Linear Predictive Coding

The efficient storage and transmission of analog signals using digital systems requires the minimization of the number of bits necessary to represent the signal while maintaining the quality to an acceptable level according to a certain criterion of performance. The conversion of an analog (continuous-time, continuous-amplitude) signal to a digital (discrete-time, discrete-amplitude) signal involves two processes: sampling and quantization. Sampling converts a continuous-time signal to a discrete-time signal by measuring its amplitude at equidistant intervals of time. Quantization involves the representation of the measured continuous amplitude by using a finite number of symbols. Therefore, a small range of amplitudes will use the same symbol (see Figure 9.4). A code word is assigned to each symbol by the coder. When the digital representation is used for digital signal processing, the quantization levels and the corresponding code words are uniformly distributed. However, for coding applications, levels may be nonuniformly distributed to match the distribution of the signal amplitudes.

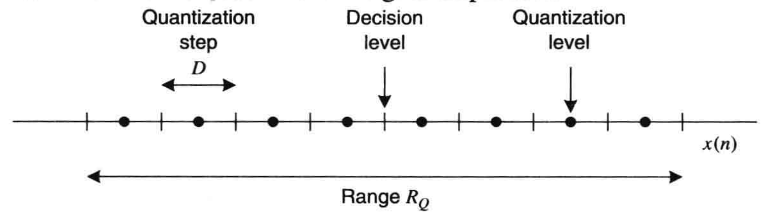


FIGURE 9.4
Partitioning of the range of a 3-bit (eight-level) uniform quantizer.

For all practical purposes, the range of a quantizer is equal to $R_Q = \Delta \cdot 2^B$, where Δ is the quantization step size

and B is the number of bits, and should cover the dynamic range of the signal. The difference between the unquantized sample $x(n)$ and the quantized sample $\hat{x}(n)$, that is,

$$e(n) \triangleq \hat{x}(n) - x(n) \tag{9.1.1}$$

is known as the *quantization error* and is always in the range $-\Delta/2 \leq e(n) \leq \Delta/2$. If we define the signal-to-noise ratio by

$$\text{SNR} \triangleq \frac{E\{x^2(n)\}}{E\{e^2(n)\}} \tag{9.1.2}$$

it can be shown (Rabiner and Schafer 1978; Jayant and Noll 1984) that

$$\text{SNR(dB)} \simeq 6B \tag{9.1.3}$$

which states that each added binary digit increases the SNR by 6 dB.

For a fixed number of bits, decreasing the dynamic range of the signal (and therefore the range of the quantizer) decreases the required quantization step and therefore the average quantization error power. Therefore, we can increase the SNR by reducing the dynamic range, or equivalently the variance of the signal. If the signal samples are significantly correlated, the variance of the difference between adjacent samples is smaller than the variance of the original signal. Thus, we can improve the SNR by quantizing this difference instead of the original signal.

The differential quantization concept is exploited by the *linear predictive coding (LPC)* system illustrated in Figure 9.5. The quantized signal is the difference

$$d(n) = x(n) - \tilde{x}(n) \tag{9.1.4}$$

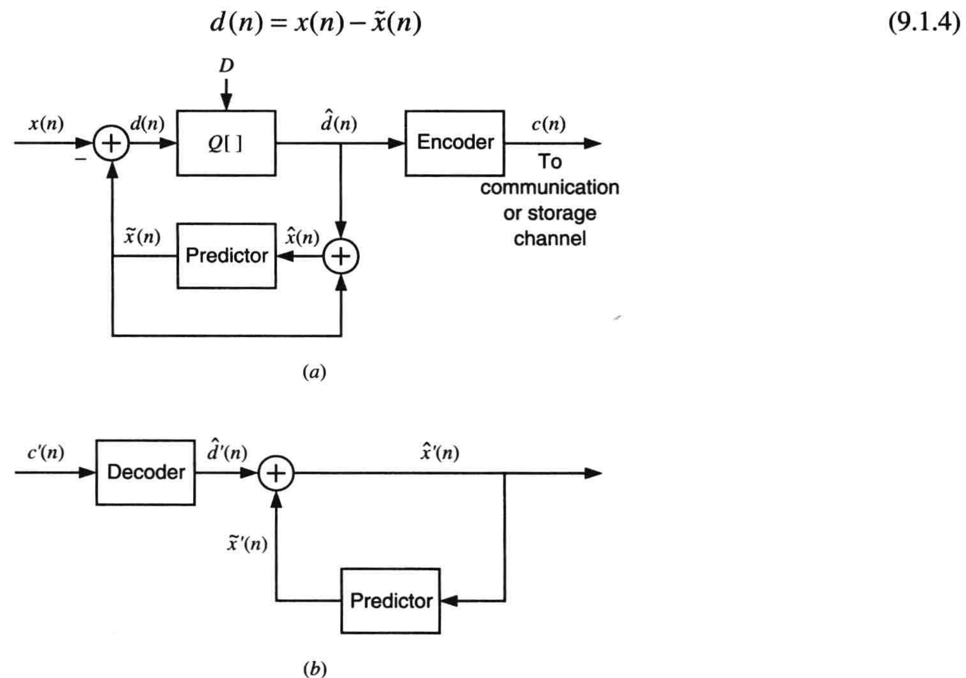


FIGURE 9.5
Block diagram of a linear predictive coding system: (*a*) coder and (*b*) decoder.

where $\tilde{x}(n)$ is an estimate or prediction of the signal $x(n)$ obtained by the predictor using a quantized version

$$\hat{x}(n) = \tilde{x}(n) + \hat{d}(n) \tag{9.1.5}$$

of the original signal (see Figure 9.5). If the quantization error of the difference signal is

$$e_d(n) = \hat{d}(n) - d(n) \tag{9.1.6}$$

we obtain

$$\hat{x}(n) = x(n) + e_d(n) \tag{9.1.7}$$

using(9.1.4) and (9.1.5). The significance of (9.1.7) is that the quantization error of the original signal is equal to the quantization error of the difference signal, independently of the properties of the predictor. Note that if $c'(n) = c(n)$,

that is , there are no transmission or storage errors, then the signal reconstructed by the decoder is $\hat{x}'(n) = \hat{x}(n)$. If the prediction is good, the dynamic range of $d(n)$ should be smaller than the dynamic range of $x(n)$, resulting in a smaller quantization noise for the same number of bits or the same quantization noise with a smaller number of bits. The performance of the LPC system depends on the accuracy of the predictor. In most practical applications, we use a linear predictor that forms an estimate (prediction) $\tilde{x}(n)$ of the present sample $x(n)$ as a linear combination of the M past samples, that is,

$$\tilde{x}(n) = \sum_{k=1}^{M} a_k \hat{x}(n-k) \tag{9.1.8}$$

The coefficients $\{a_k\}_1^M$ of the linear predictor are determined by exploiting the correlation between adjacent samples of the input signal with the objective to make the prediction error as small as possible. Since the statistical properties of the signal $x(n)$ are unknown and change with time, we *cannot* design an optimum fixed predictor. The established practical solution uses an *adaptive* linear predictor that automatically adjusts its coefficients to compute a "good" prediction at each time instant. A detailed discussion of adaptive linear prediction and its application to audio, speech, and video signal coding is provided in Jayant and Noll (1984).

9.1.3 Noise Cancelation

In Section 1.4.1 we discussed the concept of active noise control using adaptive filters. We now provide a theoretical explanation for the general problem of noise canceling using multiple sensors. The principle of general noise cancelation is illustrated in Figure 9.6. The signal of interest $s(n)$ is corrupted by uncorrelated additive noise $v_1(n)$, and the combined signal $s(n)+v_1(n)$ provides what is known as *primary input*. A second sensor, located at a different point, acquires a noise $v_2(n)$ (*reference input*) that is uncorrelated with the signal $s(n)$ but correlated with the noise $v_1(n)$. If we can design a filter that provides a good estimate $\hat{y}(n)$ of the noise $v_1(n)$, by exploiting the correlation between $v_1(n)$ and $v_2(n)$, then we could recover the desired signal by subtracting $\hat{y}(n) \approx v_1(n)$ from the primary input.

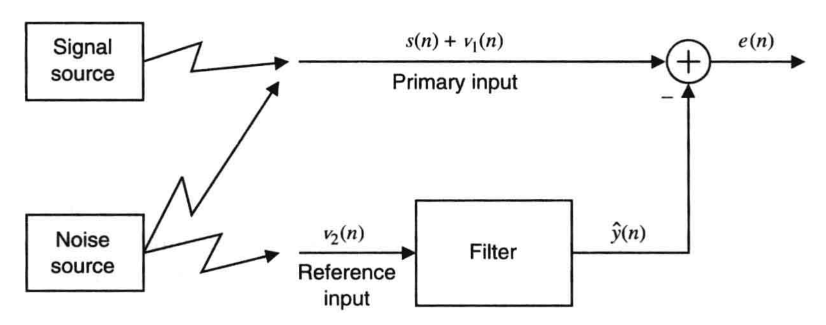


FIGURE 9.6
Principle of adaptive noise cancellation using a reference input.

Let us assume that the signals $s(n), v_1(n)$, and $v_2(n)$ are jointly wide-sense stationary with zero mean values. The "clean" signal is given by the error

$$e(n) = s(n) + [v_1(n) - \hat{y}(n)]$$

where $\hat{y}(n)$ depends on the filter structure and parameters. The MSE is given by

$$E\{|e(n)|^2\} = E\{|s(n)|^2\} + E\{|v_1(n) - \hat{y}(n)|^2\}$$

because the signals $s(n)$ and $v_1(n) - \hat{y}(n)$ are uncorrelated. Since the signal power is not influenced by the filter, if we design a filter that minimizes the total output power $E\{|e(n)|^2\}$, then that filter will minimize the output noise power $E\{|v_1(n) - \hat{y}(n)|^2\}$. Therefore, $\hat{y}(n)$ will be the MMSE estimate of the noise $v_1(n)$, and the canceler maximizes the output signal-to-noise ratio. If we know the second-order moments of the primary and reference inputs, we can design an optimum linear canceler using the techniques discussed in Chapter 5. However, in practice, the design of an optimum canceler is not feasible because the required statistical moments are either unknown or time-varying. Once again, a successful solution can be obtained by using an adaptive filter that automatically adjusts its parameters to obtain the best possible estimate of the interfering noise (Widrow et al. 1975).

9.2 Principles of Adaptive Filters

In this section, we discuss a mathematical framework for the analysis and performance evaluation of adaptive algorithms. The goal is to develop design guidelines for the application of adaptive algorithms to practical problems. The need for adaptive filters and representative applications that can benefit from their use have been discussed in Sections 1.4.1 and 9.1.

9.2.1 Features of Adaptive Filters

The applications we have discussed are only a sample from a multitude of practical problems that can be successfully solved by using adaptive filters, that is, filters that automatically change their characteristics to attain the right response at the right time. Every adaptive filtering application involves one or more input signals and a desired response signal that may or may not be accessible to the adaptive filter. We collectively refer to these signals as the *signal operating environment* (*SOE*) of the adaptive filter. Every adaptive filter consists of three modules (see Figure 9.7):

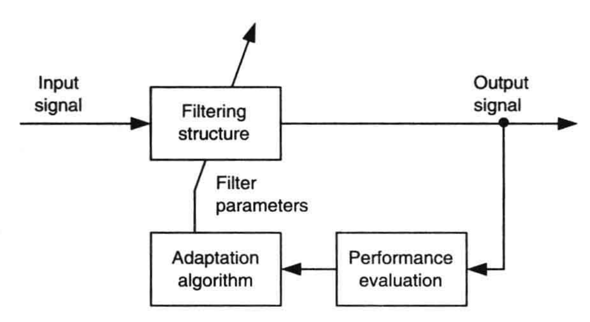


FIGURE 9.7
Basic elements of a general adaptive filter.

Filtering structure. This module forms the output of the filter using measurements of the input signal or signals. The filtering structure is *linear* if the output is obtained as a linear combination of the input measurements; otherwise it is said to be *nonlinear*. For example, the filtering module can be an adjustable finite impulse response (FIR) digital filter implemented with a direct or lattice structure or a recursive filter implemented using a cascade structure. The structure is fixed by the designer, and its parameters are adjusted by the adaptive algorithm.

Criterion of performance (COP). The output of the adaptive filter and the desired response (when available) are processed by the COP module to assess its quality with respect to the requirements of the particular application. The choice of the criterion is a balanced compromise between what is acceptable to the user of the application and what is mathematically tractable; that is, it can be manipulated to derive an adaptive algorithm. Most adaptive filters use some average form of the square error because it is mathematically tractable and leads to the design of useful practical systems.

Adaptation algorithm. The adaptive algorithm uses the value of the criterion of performance, or some function of it, and the measurements of the input and desired response (when available) to decide how to modify the parameters of the filter to improve its performance. The complexity and the characteristics of the adaptive algorithm are functions of the filtering structure and the criterion of performance.

The design of any adaptive filter requires some generic *a priori information* about the SOE and a deep understanding of the particular application. This information is needed by the designer to choose the criterion of performance and the filtering structure. Clearly, unreliable a priori information and/or incorrect assumptions about the SOE can lead to serious performance degradations or even unsuccessful adaptive filter applications. The conversion of the performance assessment to a successful parameter adjustment strategy, that is, the design of an adaptive algorithm, is the most difficult step in the design and application of adaptive filters.

If the characteristics of the SOE are constant, the goal of the adaptive filter is to find the parameters that give the

best performance and then stop the adjustment. The initial period, from the time the filter starts its operation until the time it gets reasonably close to its best performance, is known as the *acquisition* or *convergence mode*. However, when the characteristics of the SOE change with time, the adaptive filter should first find and then continuously readjust its parameters to track these changes. In this case, the filter starts with an acquisition phase that is followed by a *tracking mode*.

A very influential factor in the design of adaptive algorithms is the availability of a desired response signal. We have seen that for certain applications, the desired response may not be available for use by the adaptive filter. Therefore, the adaptation must be performed in one of two ways:

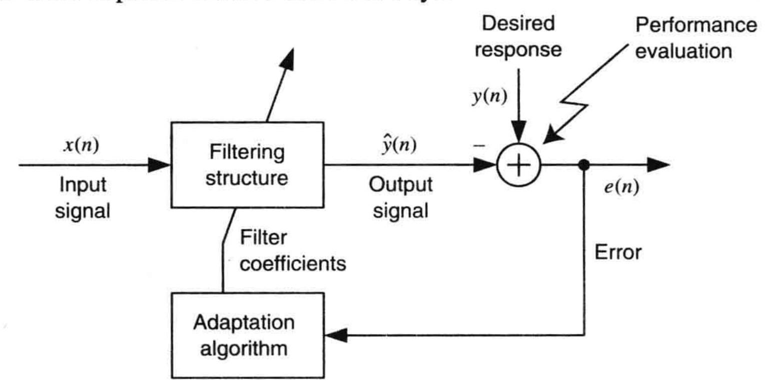


FIGURE 9.8
Basic elements of a supervised adaptive filter.

Supervised adaptation. At each time instant, the adaptive filter knows in advance the desired response, computes the error (i.e., the difference between the desired and actual response), evaluates the criterion of performance, and uses it to adjust its coefficients. In this case, the structure in Figure 9.7 is simplified to that of Figure 9.8.

Unsupervised adaptation. When the desired response is unavailable, the adaptive filter cannot explicitly form and use the error to improve its behavior. In some applications, the input signal has some measurable property (i.e., constant envelope) that is lost by the time it reaches the adaptive filter. The adaptive filter adjusts its parameters in such a way as to restore the lost property of the input signal. The *property restoral approach* to adaptive filtering was introduced in Treichler et al. (1987). In some other applications (e.g., digital communications) the basic task of the adaptive filter is to classify each received pulse to one of a finite set of symbols. In this case we basically have a problem of unsupervised classification (Fukunaga 1990).

In this chapter we focus our discussion on supervised adaptive filters, that is, filters that have access to a desired response signal.

9.2.2 Optimum versus Adaptive Filters

We have mentioned several times that the theory of stochastic processes provides the mathematical framework for the design and analysis of optimum filters. In Chapter 5, we introduced filters that are optimum according to the MSE criterion of performance; and in Chapter 6, we developed algorithms and structures for their efficient design and implementation. However, optimum filters are a theoretical tool and cannot be used in practical applications because we do not know the statistical quantities (e.g., second-order moments) that are required for their design. Adaptive filters can be thought as the practical counterpart of optimum filters: They try to reach the performance of optimum filters by processing measurements of the SOE in real time, which makes up for the lack of a priori statistics.

For this analysis, we consider the general case of a linear combiner that includes filtering and prediction as special cases. However, for convenience we use the terms *filters* and *filtering*. We remind the reader that, from a mathematical point of view, the key difference between a linear combiner and an FIR filter or predictor is the shift invariance (temporal ordering) of the input data vector. This difference, which is illustrated in Figure 9.9, also has important implications in the implementation of adaptive filters. To this end, suppose that the SOE is comprised of M

input signals $x_k(n,\zeta)$ and a desired response signal $y(n,\zeta)$, which are sample realizations of random sequences.[1]

Then the estimate of $y(n,\zeta)$ is computed by using the linear combiner

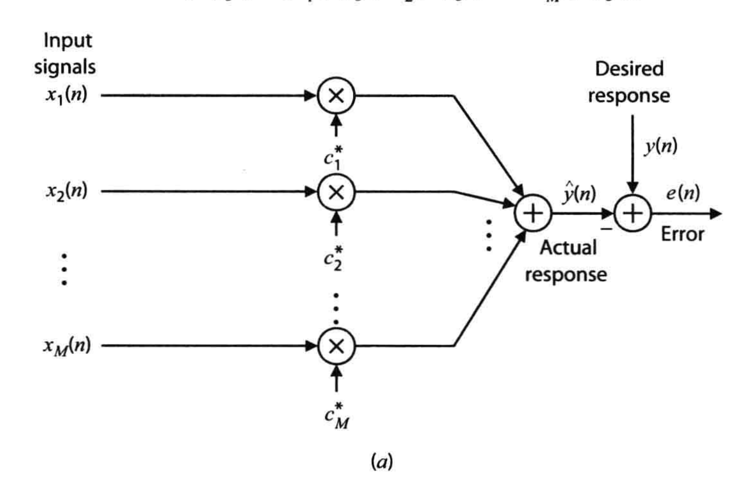
$$\hat{y}(n,\zeta)=\sum_{k=1}^{M} c_k^*(n)x_k(n,\zeta) \triangleq \boldsymbol{c}^H(n)\boldsymbol{x}(n,\zeta) \tag{9.2.1}$$

where

$$\boldsymbol{c}(n)=[c_1(n)\ c_2(n)\ \cdots\ c_M(n)]^{\mathrm{T}} \tag{9.2.2}$$

is the coefficient vector and

$$\boldsymbol{x}(n,\zeta)=[x_1(n,\zeta)\ x_2(n,\zeta)\ \cdots\ x_M(n,\zeta)]^{\mathrm{T}} \tag{9.2.3}$$



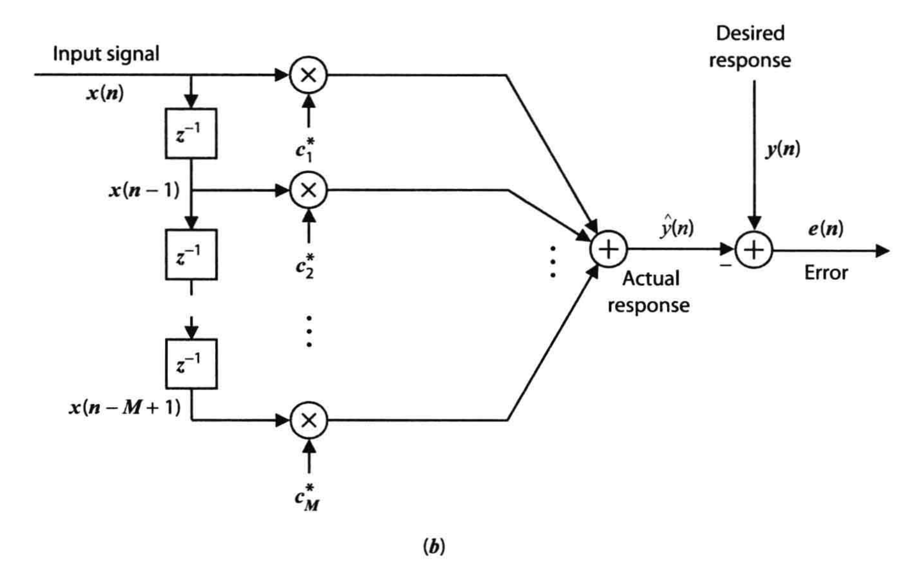


FIGURE 9.9

Illustration of the difference of the input signal between (*a*) a multiple-input linear combiner and (*b*) a single-input FIR filter.

is the input data vector. For single-sensor applications, the input data vector is shift-invariant

$$\boldsymbol{x}(n)=[x(n,\zeta)\ x(n-1,\zeta)\ \cdots\ x(n-M+1,\zeta)]^{\mathrm{T}} \tag{9.2.4}$$

and the linear combiner takes the form of the FIR filter

[1] For clarity, in this section only, we include the dependence on ζ to denote random variables.

$$\hat{y}(n,\zeta)=\sum_{k=0}^{M-1}h(n,k)x(n-k,\zeta)\triangleq c^{H}(n)x(n,\zeta) \tag{9.2.5}$$

where $c_k(n)=h^*(n,k)$ are the samples of the impulse response at time n.

Optimum filters. If we know the second-order moments of the SOE, we can design an optimum filter $\mathbf{c}_o(n)$ by solving the normal equations

$$R(n)c_o(n)=d(n) \tag{9.2.6}$$

where
$$R(n)=E\{x(n,\zeta)x^{H}(n,\zeta)\} \tag{9.2.7}$$

and
$$d(n)=E\{x(n,\zeta)y^*(n,\zeta)\} \tag{9.2.8}$$

are the correlation matrix of the input data vector and the cross-correlation between the input data vector and the desired response, respectively. During its normal operation, the optimum filter works with specific realizations of the SOE, that is,

$$\hat{y}_o(n,\zeta)=c_o^{H}(n)x(n,\zeta) \tag{9.2.9}$$

$$\varepsilon_o(n,\zeta)=y(n,\zeta)-\hat{y}_o(n,\zeta) \tag{9.2.10}$$

where $\hat{y}_o(n,\zeta)$ is the optimum estimate and $\varepsilon_o(n,\zeta)$ is the optimum instantaneous error [see Figure 9.10(*a*)]. However, the filter is optimized with respect to its average performance across all possible realizations of the SOE, and the MMSE

$$P_o(n)=E\{|\varepsilon_o(n,\zeta)|^2\}=P_y(n)-d^{H}(n)c_o(n) \tag{9.2.11}$$

shows how well the filter performs on average. Also, we emphasize that the optimum coefficient vector is a nonrandom quantity and that the desired response is not essential for the operation of the optimum filter [see Equation (9.2.9)].

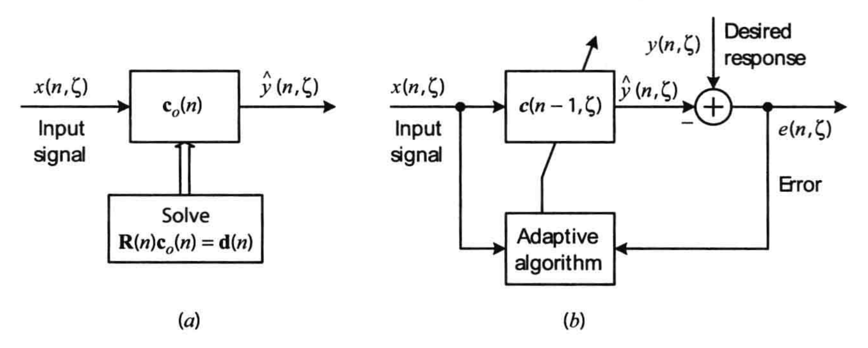


FIGURE 9.10
Illustration of the difference in operation between (*a*) optimum filters and (*b*) adaptive filters.

If the SOE is stationary, the optimum filter is computed once and is used with *all* realizations $\{\mathbf{x}(n,\zeta),y(n,\zeta)\}$. For nonstationary environments, the optimum filter design is repeated at every time instant n because the optimum filter is time-varying.

Adaptive filters. In most practical applications, where the second-order moments $R(n)$ and $d(n)$ are *unknown*, the use of an adaptive filter is the best solution. If the SOE is ergodic, we have

$$R=\lim_{N\to\infty}\frac{1}{2N+1}\sum_{n=-N}^{N}x(n,\zeta)x^{H}(n,\zeta) \tag{9.2.12}$$

$$d=\lim_{N\to\infty}\frac{1}{2N+1}\sum_{n=-N}^{N}x(n,\zeta)y^*(n,\zeta) \tag{9.2.13}$$

because ensemble averages are equal to time averages (see Section 2.1). If we collect a sufficient amount of data $\{\mathbf{x}(n,\zeta), y(n,\zeta)\}_0^{N-1}$, we can obtain an acceptable estimate of the optimum filter by computing the estimates

$$\hat{\boldsymbol{R}}_N(\zeta) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{x}(n,\zeta)\boldsymbol{x}^{\mathrm{H}}(n,\zeta) \tag{9.2.14}$$

$$\hat{\boldsymbol{d}}_N(\zeta) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{x}(n,\zeta)y^*(n,\zeta) \tag{9.2.15}$$

by time-averaging and then solving the linear system

$$\hat{\boldsymbol{R}}_N(\zeta)\boldsymbol{c}_N(\zeta) = \hat{\boldsymbol{d}}_N(\zeta) \tag{9.2.16}$$

The obtained coefficients can be used to filter the data in the interval $0 \le n \le N-1$ or to start filtering the data for $n \ge N$, on a sample-by-sample basis, in real time. This procedure, which we called *block adaptive filtering* in Chapter 7, should be repeated each time the properties of the SOE change significantly. Clearly, block adaptive filters cannot track statistical variations within the operating block and cannot be used in all applications.

Indeed, there are applications, for example, adaptive equalization, in which each input sample should be processed immediately after its observation and before the arrival of the next sample. In such cases, we should use a sample-by-sample adaptive filter that starts filtering immediately after the observation of the pair $\{\boldsymbol{x}(0), y(0)\}$ using a "guess" $\boldsymbol{c}(-1)$ for the adaptive filter coefficients. Usually, the initial guess $\boldsymbol{c}(-1)$ is a very poor estimate of the optimum filter $\boldsymbol{c}_o$. However, this estimate is improved with time as the filter processes additional pairs of observations.

As we discussed in Section 9.2.1, an adaptive filter consists of three key modules: an adjustable filtering structure that uses input samples to compute the output, the criterion of performance that monitors the performance of the filter, and the adaptive algorithm that updates the filter coefficients. The key component of any adaptive filter is the *adaptive algorithm*, which is a rule to determine the filter coefficients from the available data $\boldsymbol{x}(n,\zeta)$ and $y(n,\zeta)$ [see Figure 9.10(*b*)]. The dependence of $\boldsymbol{c}(n,\zeta)$ on the input signal makes the adaptive filter a nonlinear and time-varying stochastic system.

The data available to the adaptive filter at time *n* are the input data vector $\boldsymbol{x}(n,\zeta)$, the desired response $y(n,\zeta)$, and the most recent update $\boldsymbol{c}(n-1,\zeta)$ of the coefficient vector. The adaptive filter, at each time n, performs the following computations:

1. *Filtering:*

$$\hat{y}(n,\zeta) = \boldsymbol{c}^{\mathrm{H}}(n-1,\zeta)\boldsymbol{x}(n,\zeta) \tag{9.2.17}$$

2. *Error formation:*

$$e(n,\zeta) = y(n,\zeta) - \hat{y}(n,\zeta) \tag{9.2.18}$$

3. *Adaptive algorithm:*

$$\boldsymbol{c}(n,\zeta) = \boldsymbol{c}(n-1,\zeta) + \Delta\boldsymbol{c}\{\boldsymbol{x}(n,\zeta), e(n,\zeta)\} \tag{9.2.19}$$

where the increment or correction term $\Delta\boldsymbol{c}(n,\zeta)$ is chosen to bring $\boldsymbol{c}(n,\zeta)$ close to $\boldsymbol{c}_o$, with the passage of time. If we can successively determine the corrections $\Delta\boldsymbol{c}(n,\zeta)$ so that $\boldsymbol{c}(n,\zeta) \simeq \boldsymbol{c}_o$, that is, $\|\boldsymbol{c}(n,\zeta) - \boldsymbol{c}_o\| < \delta$, for some $n > N_\delta$, we obtain a good approximation for $\boldsymbol{c}_o$ by avoiding the explicit averagings (9.2.14), (9.2.15), and the solution of the normal equations (9.2.16). A key requirement is that $\Delta\boldsymbol{c}(n,\zeta)$ must vanish if the error $e(n,\zeta)$ vanishes. Hence, $e(n,\zeta)$ plays a major role in determining the increment $\Delta\boldsymbol{c}(n,\zeta)$.

We notice that the estimate $\hat{y}(n,\zeta)$ of the desired response $y(n,\zeta)$ is evaluated using the *current* input vector $\boldsymbol{x}(n,\zeta)$ and the *past* filter coefficients $\boldsymbol{c}(n-1,\zeta)$. The estimate $\hat{y}(n,\zeta)$ and the corresponding error $e(n,\zeta)$ can be considered as *predicted* estimates compared to the *actual* estimates that would be evaluated using the *current* coefficient vector $\boldsymbol{c}(n,\zeta)$. Coefficient updating methods that use the predicted error $e(n,\zeta)$ are known as *a priori type adaptive algorithms*.

If we use the *actual* estimates, obtained using the current estimate $\boldsymbol{c}(n,\zeta)$ of the adaptive filter coefficients, we have

1. *Filtering:*

$$\hat{y}_a(n,\zeta) = \mathbf{c}^H(n,\zeta)\mathbf{x}(n,\zeta) \tag{9.2.20}$$

2. *Error formation:*

$$\varepsilon(n,\zeta) = y(n,\zeta) - \hat{y}_a(n,\zeta) \tag{9.2.21}$$

3. *Adaptive algorithm:*

$$\mathbf{c}(n,\zeta) = \mathbf{c}(n-1,\zeta) + \Delta\mathbf{c}\{\mathbf{x}(n,\zeta), \varepsilon(n,\zeta)\} \tag{9.2.22}$$

which are known as *a posteriori type adaptive algorithms*. The terms *a priori* and *a posteriori* were introduced in Carayannis et al. (1983) to emphasize the use of estimates evaluated before or after the updating of the filter coefficients. The difference between a priori and a posteriori errors and their meanings will be further clarified when we discuss adaptive least-squares filters in Section 9.5. The timing diagram for the above two algorithms is shown in Figure 9.11.

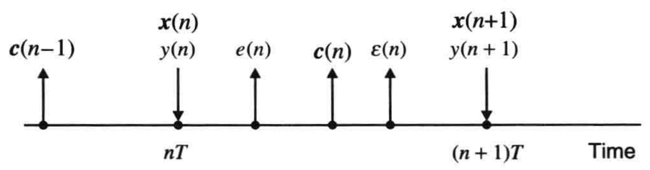


FIGURE 9.11
Timing diagrams for a priori and a posteriori adaptive algorithms.

In conclusion, the objective of an adaptive filter is to use the available data at time n, namely, $\{\mathbf{x}(n,\zeta),\ y(n,\zeta),\ \mathbf{c}(n-1,\zeta)\}$, to update the "old" coefficient vector $\mathbf{c}(n-1,\zeta)$ to a "new" estimate $\mathbf{c}(n,\zeta)$ so that $\mathbf{c}(n,\zeta)$ is closer to the optimum filter vector $\mathbf{c}_o(n)$ and the output $\hat{y}(n)$ is a better estimate of the desired response $y(n)$. Most adaptive algorithms have the following form:

$$\begin{bmatrix}\text{New coefficient}\\ \text{vector}\end{bmatrix} = \begin{bmatrix}\text{old coefficient}\\ \text{vector}\end{bmatrix} + \begin{bmatrix}\text{adaptation gain}\\ \text{vector}\end{bmatrix} \cdot \begin{pmatrix}\text{error}\\ \text{signal}\end{pmatrix} \tag{9.2.23}$$

where the error signal is the difference between the desired response and the predicted or actual outputs of the adaptive filter. One of the fundamental differences among the various algorithms is the optimality of the used adaptation gain vector and the amount of computation required for its evaluation.

9.2.3 Stability and Steady-State Performance of Adaptive Filters

We now address the issues of stability and performance of adaptive filters. Since the goal of an adaptive filter $\mathbf{c}(n,\zeta)$ is *first to find and then track* the optimum filter $\mathbf{c}_o(n)$ as *quickly and accurately* as possible, we can evaluate its performance by measuring some function of its deviation

$$\tilde{\mathbf{c}}(n,\zeta) \triangleq \mathbf{c}(n,\zeta) - \mathbf{c}_o(n) \tag{9.2.24}$$

from the corresponding optimum filter. Clearly, an acceptable adaptive filter should be stable in the bounded-input bounded-output (BIBO) sense, and its performance should be close to that of the associated optimum filter. The analysis of BIBO stability is extremely difficult because adaptive filters are nonlinear, time-varying systems working in a random SOE. The performance of adaptive filters is primarily measured by investigating the value of the MSE as a function of time. To discuss these problems, first we consider an adaptive filter working in a stationary SOE, and then we extend our discussion to a nonstationary SOE.

Stability

The adaptive filter starts its operation at time, say, $n = 0$, and by processing the observations $\{x(n,\zeta), y(n,\zeta)\}_0^\infty$ generates a sequence of vectors $\{\mathbf{c}(n,\zeta)\}_0^\infty$ using the adaptive algorithm. Since the FIR filtering structure is always stable, the output or the error of the adaptive filter will be bounded if its coefficients are always kept close to the coefficients of the associated optimum filter. However, the presence of the feedback loop in every adaptive filter (see Figure 9.10) raises the issue of stability. In a stationary SOE, where the optimum filter $\mathbf{c}_o$ is constant, convergence of $\mathbf{c}(n,\zeta)$ to $\mathbf{c}_o$ as $n \to \infty$ will guarantee the BIBO stability of the adaptive filter. For a

specific realization ζ, the kth component $c_k(n,\zeta)$ or the norm $\|c(n,\zeta)\|$ of the vector $c(n,\zeta)$ is a sequence of numbers that might or might not converge.[①] Since the coefficients $c_k(n,\zeta)$ are random, we must use the concept of *stochastic convergence* (Papoulis 1991).

We say that a random sequence converges *everywhere* if the sequence $c_k(n,\zeta)$ converges for every ζ, that is,

$$\lim_{n\to\infty} c_k(n,\zeta) = c_{o,k}(\zeta) \tag{9.2.25}$$

where the limit $c_{o,k}(\zeta)$ depends, in general, on ζ. Requiring the adaptive filter to converge to $\boldsymbol{c}_o$ for every possible realization of the SOE is both hard to guarantee and not necessary, because some realizations may have very small or zero probability of occurrence.

If we wish to ensure that the adaptive filter converges for the realizations of the SOE that may actually occur, we can use the concept of convergence almost everywhere. We say that the random sequence $c_k(n,\zeta)$ *converges almost everywhere* or *with probability 1* if

$$P\{\lim_{n\to\infty} |c_k(n,\zeta) - c_{o,k}(\zeta)| = 0\} = 1 \tag{9.2.26}$$

which implies that there can be some sample sequences that do not converge, which must occur with probability zero.

Another type of stochastic convergence that is used in adaptive filtering is defined by

$$\lim_{n\to\infty} E\{|c_k(n,\zeta) - c_{o,k}|^2\} = \lim_{n\to\infty} E\{|\tilde{c}_k(n,\zeta)|^2\} = 0 \tag{9.2.27}$$

and is known as *convergence in the MS sense*. The primary reason for the use of mean square (MS) convergence is that unlike the almost-everywhere convergence, it uses only one sequence of numbers that takes into account the averaging effect of all sample sequences. Furthermore, it uses second-order moments for verification and has an interpretation in terms of power. Convergence in MS does not imply—nor is implied by—convergence with probability 1. Since

$$\frac{E\{|\tilde{c}_k(n,\zeta)|^2\}}{\delta} = \frac{|E\{\tilde{c}_k(n,\zeta)\}|^2}{\delta} + \frac{var\{\tilde{c}_k(n,\zeta)\}}{\delta^2} \tag{9.2.28}$$

if we can show that $E\{\tilde{c}_k(n)\} \to 0$ as $n \to \infty$ and $var\{\tilde{c}_k(n,\zeta)\}$ is bounded for all n, we can ensure convergence in MS. In this case, we can say that an adaptive filter that operates in a stationary SOE is an asymptotically stable filter.

Performance measures

In theoretical investigations, any quantity that measures the deviation of an adaptive filter from the corresponding optimum filter can be used to evaluate its performance.

The *mean square deviation (MSD)*

$$\mathscr{D}(n) \triangleq E\{\|c(n,\zeta) - c_o(n)\|^2\} = E\{\|\tilde{c}(n,\zeta)\|^2\} \tag{9.2.29}$$

measures the average distance between the coefficient vectors of the adaptive and optimum filters. Although the MSD is not measurable in practice, it is useful in analytical studies. Adaptive algorithms that minimize $\mathscr{D}(n)$ for each value of n are known as algorithms with *optimum learning*.

In Section 5.2.2 we showed that if the input correlation matrix is positive definite, any deviation, say, $\tilde{c}(n)$, of the optimum filter coefficients from their optimum setting increases the mean square error (MSE) by an amount equal to $\tilde{c}^H(n)\boldsymbol{R}\tilde{c}(n)$, known as *excess MSE (EMSE)*. In adaptive filters, the random deviation $\tilde{c}(n,\zeta)$ from the optimum results in an EMSE, which is measured by the ensemble average of $\tilde{c}^H(n,\zeta)\boldsymbol{R}\tilde{c}(n,\zeta)$. For a posteriori adaptive filters, the MSE can be decomposed as

$$P'(n) \triangleq E\{|\varepsilon(n,\zeta)|^2\} \triangleq P'_o(n) + P'_{ex}(n) \tag{9.2.30}$$

where $P'_{ex}(n)$ is the EMSE and $P'_o(n)$ is the MMSE given by

---

[①] We recall that a sequence of real nonrandom numbers $a_0, a_1, a_2, \ldots$ converges to a number a if and only if for every positive number δ there exists a positive integer N_δ such that for all $n > N_\delta$, we have $|a_n - a| < \delta$. This is abbreviated by $\lim_{n\to\infty} a_n = a$.

$$P_o'(n) \triangleq E\{|\varepsilon_o(n,\zeta)|^2\} \tag{9.2.31}$$

with
$$\varepsilon_o(n,\zeta) \triangleq y(n,\zeta) - \mathbf{c}_o^H(n)\mathbf{x}(n,\zeta) \tag{9.2.32}$$
as the a posteriori optimum filtering error. Clearly, the a posteriori EMSE $P_{\text{ex}}'(n)$ is given by

$$P_{\text{ex}}'(n) \triangleq P'(n) - P_o'(n) \tag{9.2.33}$$

For a priori adaptive algorithms, where we use the "old" coefficient vector $\mathbf{c}(n-1,\zeta)$, it is more appropriate to use the a priori EMSE given by

$$P_{\text{ex}}(n) \triangleq P(n) - P_o(n) \tag{9.2.34}$$

where
$$P(n) \triangleq E\{|e_o(n,\zeta)|^2\} \tag{9.2.35}$$

and
$$P_o(n) \triangleq E\{|e_o(n,\zeta)|^2\} \tag{9.2.36}$$

with
$$e_o(n,\zeta) \triangleq y(n,\zeta) - \mathbf{c}_o^H(n-1)\mathbf{x}(n,\zeta) \tag{9.2.37}$$

as the a priori optimum filtering error. If the SOE is stationary, we have $\varepsilon_o(n,\zeta) = e_o(n,\zeta)$; that is, the optimum a priori and a posteriori errors are identical.

The dimensionless ratio

$$\mathscr{M}(n) \triangleq \frac{P_{\text{ex}}(n)}{P_o(n)} \qquad \text{or} \qquad \mathscr{M}'(n) \triangleq \frac{P_{\text{ex}}'(n)}{P_o'(n)} \tag{9.2.38}$$

known as *misadjustment*, is a useful measure of the quality of adaptation. Since the EMSE is always positive, *there is no adaptive filter that can perform (on the average) better than the corresponding optimum filter*. In this sense, we can say that the excess MSE or the misadjustment measures the cost of adaptation.

Acquisition and tracking

Plots of the MSD, MSE, or $\mathscr{M}(n)$ as a function of n, which are known as *learning curves*, characterize the performance of an adaptive filter and are widely used in theoretical and experimental studies. When the adaptive filter starts its operation, its coefficients provide a poor estimate of the optimum filter and the MSD or the MSE is very large. As the number of observations processed by the adaptive filter increases with time, we expect the quality of the estimate $\mathbf{c}(n,\zeta)$ to improve, and therefore the MSD and the MSE to decrease. The property of an adaptive filter to bring the coefficient vector $\mathbf{c}(n,\zeta)$ close to the optimum filter $\mathbf{c}_o$, independently of the initial condition $\mathbf{c}(-1)$ and the statistical properties of the SOE, is called *acquisition*. During the acquisition phase, we say that the adaptive filter is in a transient mode of operation.

A natural requirement for any adaptive algorithm is that adaptation stops after the algorithm has found the optimum filter $\mathbf{c}_o$. However, owing to the randomness of the SOE and the finite amount of data used by the adaptive filter, its coefficients continuously fluctuate about their optimum settings, that is, about the coefficients of the optimum filter, in a random manner. As a result, the adaptive filter reaches a steady-state mode of operation, after a certain time, and its performance stops improving.

The transient and steady-state modes of operation in a stationary SOE are illustrated in Figure 9.12(*a*). The duration of the acquisition phase characterizes the *speed of adaptation* or *rate of convergence* of the adaptive filter, whereas the steady-state EMSE or misadjustment characterizes the *quality of adaptation*. These properties depend on the SOE, the filtering structure, and the adaptive algorithm.

At each time n, any adaptive filter computes an estimate of the optimum filter using a finite amount of data. The error resulting from the finite amount of data is known as *estimation error*. An additional error, known as the *lag error*, results when the adaptive filter attempts to track a time-varying optimum filter $\mathbf{c}_o(n)$ in a nonstationary SOE. The modes of operation of an adaptive filter in a nonstationary SOE are illustrated in Figure 9.12(*b*). The SOE of the adaptive filter becomes nonstationary if $\mathbf{x}(n,\zeta)$ or $y(n,\zeta)$ or both are nonstationary. The nonstationarity of the

input is more severe than that of the desired response because it may affect the invertibility of $\boldsymbol{R}(n)$. Since the adaptive filter has to first acquire and then track the optimum filter, tracking is a steady-state property. Therefore, in general, the speed of adaptation (a transient-phase property) and the tracking capability (a steady-state property) are two different characteristics of the adaptive filter. Clearly, tracking is feasible only if the statistics of the SOE change "slowly" compared to the speed of tracking of the adaptive filter. These concepts will become more precise in Section 9.8, where we discuss the tracking properties of adaptive filters.

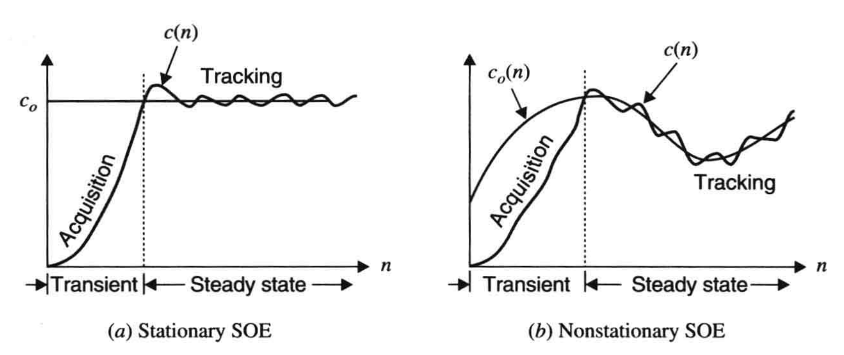


FIGURE 9.12
Modes of operation in a stationary and nonstationary SOE.

9.2.4 *Some Practical Considerations*

The complexity of the hardware or software implementation of an adaptive filter is basically determined by the following factors: (1) the number of instructions per time update or computing time required to complete one time updating; (2) the number of memory locations required to store the data and the program instructions; (3) the structure of information flow in the algorithm, which is very important for implementations using parallel processing, systolic arrays, or VLSI chips; and (4) the investment in hardware design tools and software development. We focus on implementations for general-purpose computers or special-purpose digital signal processors that basically involve programming in a high level or assembly language. More details about DSP software development can be found in Embree and Kimble (1991) and in Lapsley et al. (1997).

The digital implementation of adaptive filters implies the use of finite-word-length arithmetic. As a result, the performance of the practical (finite-precision) adaptive filters deviates from the performance of ideal (infinite-precision) adaptive filters. Finite-precision implementation affects the performance of adaptive filters in several complicated ways. The major factors are (1) the quantization of the input signal(s) and the desired response, (2) the quantization of filter coefficients, and (3) the roundoff error in the arithmetic operations used to implement the adaptive filter. The nonlinear nature of adaptive filters coupled with the nonlinearities introduced by the finite-word-length arithmetic makes the performance evaluation of practical adaptive filters extremely difficult. Although theoretical analysis provides insight and helps to clarify the behavior of adaptive filters, the most effective way is to simulate the filter and measure its performance.

Finite precision affects two important properties of adaptive filters, which, although related, are not equivalent. Let us denote by $\boldsymbol{c}_{\text{ip}}(n)$ and $\boldsymbol{c}_{\text{fp}}(n)$ the coefficient vectors of the filter implemented using infinite- and finite-precision arithmetic, respectively. An adaptive filter is said to be *numerically stable* if the difference vector $\boldsymbol{c}_{\text{ip}}(n)-\boldsymbol{c}_{\text{fp}}(n)$ remains always bounded, that is, the roundoff error propagation system is stable. Numerical stability is an inherent property of the adaptive algorithm and cannot be altered by increasing the numerical precision. Indeed, increasing the word length or reorganizing the computations will simply delay the divergence of an adaptive filter; only actual change of the algorithm can stabilize an adaptive filter by improving the properties of the roundoff error propagation system (Ljung and Ljung 1985; Cioffi 1987).

The *numerical accuracy* of an adaptive filter measures the deviation, at steady state, of any obtained estimates from theoretically expected values, due to roundoff errors. Numerical accuracy results in an increase of the output error without catastrophic problems and can be reduced by increasing the word length. In contrast, lack of numerical stability leads to catastrophic overflow (divergence or blowup of the algorithm) as a result of roundoff error accumulation. Numerically unstable algorithms converging before "explosion" may provide good numerical accuracy. Therefore, although the two properties are related, one does not imply the other.

Two other important issues are the sensitivity of an algorithm to bad or abnormal input data (e.g., poorly exciting input) and its sensitivity to initialization. All these issues are very important for the application of adaptive algorithms to real-world problems and are further discussed in the context of specific algorithms.

9.3 Method of Steepest Descent

Most adaptive filtering algorithms are obtained by simple modifications of iterative methods for solving deterministic optimization problems. Studying these techniques helps one to understand several aspects of the operation of adaptive filters. In this section we discuss gradient-based optimization methods because they provide the ground for the development of the most widely used adaptive filtering algorithms.

As we discussed in Section 5.2.1, the error performance surface of an optimum filter, in a stationary SOE, is given by

$$P(\boldsymbol{c}) = P_y - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{d} - \boldsymbol{d}^{\mathrm{H}}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}}\boldsymbol{R}\boldsymbol{c} \tag{9.3.1}$$

where $P_y = E\{|y(n)|^2\}$. Equation (9.3.1) is a quadratic function of the coefficients and represents a bowl-shaped surface (when $\boldsymbol{R}$ is positive definite) and has a unique minimum at $\boldsymbol{c}_o$ (optimum filter). There are two distinct ways to find the minimum of (9.3.1):

1. Solve the normal equations $\boldsymbol{R}\boldsymbol{c} = \boldsymbol{d}$, using a direct linear system solution method.
2. Find the minimum of $P(\boldsymbol{c})$, using an iterative minimization algorithm.

Although direct methods provide the solution in a finite number of steps, sometimes we prefer iterative methods because they require less numerical precision, are computationally less expensive, work when $\boldsymbol{R}$ is not invertible, and are the only choice for nonquadratic performance functions.

In all iterative methods, we start with an approximate solution (a guess), which we keep changing until we reach the minimum. Thus, to find the optimum $\boldsymbol{c}_o$, we start at some arbitrary point $\boldsymbol{c}_0$, usually the null vector $\boldsymbol{c}_0 = 0$, and then start a search for the "bottom of the bowl." The key is to choose the steps in a systematic way so that each step takes us to a lower point until finally we reach the bottom. What differentiates various optimization algorithms is how we choose the direction and the size of each step.

Steepest-descent algorithm (SDA)

If the function $P(\boldsymbol{c})$ has continuous derivatives, it is possible to approximate its value at an arbitrary neighboring point $\boldsymbol{c} + \Delta\boldsymbol{c}$ by using the Taylor expansion

$$P(\boldsymbol{c} + \Delta\boldsymbol{c}) = P(\boldsymbol{c}) + \sum_{i=1}^{M} \frac{\partial P(\boldsymbol{c})}{\partial c_i} \Delta c_i + \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \Delta c_i \frac{\partial^2 P(\boldsymbol{c})}{\partial c_i \partial c_j} \Delta c_j + \cdots \tag{9.3.2}$$

or more compactly

$$P(\boldsymbol{c} + \Delta\boldsymbol{c}) = P(\boldsymbol{c}) + (\Delta\boldsymbol{c})^{\mathrm{T}} \nabla P(\boldsymbol{c}) + \frac{1}{2} (\Delta\boldsymbol{c})^{\mathrm{T}} [\nabla^2 P(\boldsymbol{c})](\Delta\boldsymbol{c}) + \cdots \tag{9.3.3}$$

where $\nabla P(\boldsymbol{c})$ is the gradient vector, with elements $\partial P(\boldsymbol{c})/\partial c_i$, and $\nabla^2 P(\boldsymbol{c})$ is the Hessian matrix, with elements $\partial^2 P(\boldsymbol{c})/(\partial c_i \partial c_j)$. For simplicity we consider filters with real coefficients, but the conclusions apply when the coefficients are complex. For the quadratic function (9.3.1), we have

$$\nabla P(\boldsymbol{c}) = 2(\boldsymbol{R}\boldsymbol{c} - \boldsymbol{d}) \tag{9.3.4}$$

$$\nabla^2 P(\boldsymbol{c}) = 2\boldsymbol{R} \tag{9.3.5}$$

and the higher-order terms are zero. For nonquadratic functions, higher-order terms are nonzero, but if $\|\Delta\boldsymbol{c}\|$ is small, we can use a quadratic approximation. We note that if $\nabla P(\boldsymbol{c}_o) = 0$ and $\boldsymbol{R}$ is positive definite, then $\boldsymbol{c}_o$ is the minimum because $(\Delta\boldsymbol{c})^{\mathrm{T}}[\nabla^2 P(\boldsymbol{c}_o)] \cdot (\Delta\boldsymbol{c}) > 0$ for any nonzero $\Delta\boldsymbol{c}$. Hence, if we choose the step $\Delta\boldsymbol{c}$ so that $(\Delta\boldsymbol{c})^{\mathrm{T}} \nabla P(\boldsymbol{c}) < 0$, we will have $P(\boldsymbol{c} + \Delta\boldsymbol{c}) < P(\boldsymbol{c})$, that is, we make a step to a point closer to the minimum. Since $(\Delta\boldsymbol{c})^{\mathrm{T}} \nabla P(\boldsymbol{c}) = \|\Delta\boldsymbol{c}\| \|\nabla P(\boldsymbol{c})\| \cos\theta$, the reduction in MSE is maximum when $\Delta\boldsymbol{c} = -\nabla P(\boldsymbol{c})$. For this reason, the direction of the negative gradient is known as the direction of *steepest descent*. This leads to the following iterative

minimization algorithm

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + \mu[-\nabla P(\boldsymbol{c}_{k-1})] \qquad k \geq 0 \tag{9.3.6}$$

which is known as the *method of steepest descent* (Scales 1985). The positive constant μ, known as the *step-size parameter*, controls the size of the descent in the direction of the negative gradient. The algorithm is usually initialized with $\boldsymbol{c}_0 = 0$. The steepest-descent algorithm (SDA) is illustrated in Figure 9.13 for a single-parameter case.

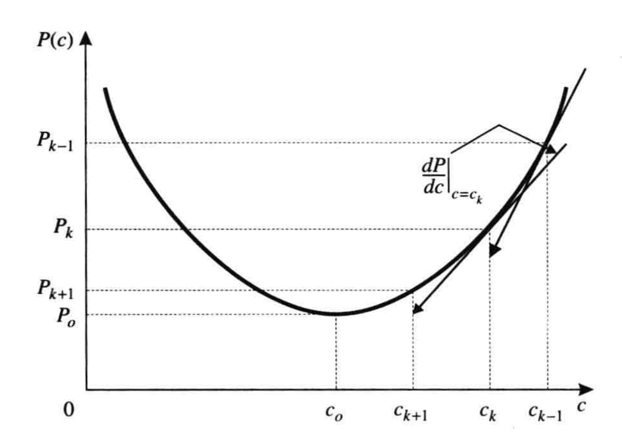


FIGURE 9.13
Illustration of gradient search of the MSE surface for the minimum error point.

For the cost function in (9.3.1), the SDA becomes

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + 2\mu(\boldsymbol{d} - \boldsymbol{R}\boldsymbol{c}_{k-1}) = (\boldsymbol{I} - 2\mu\boldsymbol{R})\boldsymbol{c}_{k-1} + 2\mu\boldsymbol{d} \tag{9.3.7}$$

which is a recursive difference equation. Note that *k* denotes an iteration in the SDA and has nothing to do with time. However, this iterative optimization can be combined with filtering to obtain a type of "asymptotically" optimum filter defined by

$$e(n,\zeta) = y(n,\zeta) - \boldsymbol{c}_{n-1}^{\mathrm{H}}\boldsymbol{x}(n,\zeta) \tag{9.3.8}$$

$$\boldsymbol{c}_n = \boldsymbol{c}_{n-1} + 2\mu(\boldsymbol{d} - \boldsymbol{R}\boldsymbol{c}_{n-1}) \tag{9.3.9}$$

and is further discussed in Problem 9.2.

There are two key performance factors in the design of iterative optimization algorithms: stability and rate of convergence.

Stability

An algorithm is said to be *stable* if it converges to the minimum regardless of the starting point. To investigate the stability of SDA, we rewrite (9.3.7) in terms of the coefficient error vector

$$\tilde{\boldsymbol{c}}_k \triangleq \boldsymbol{c}_k - \boldsymbol{c}_o \qquad k \geq 0 \tag{9.3.10}$$

as

$$\tilde{\boldsymbol{c}}_k = (\boldsymbol{I} - 2\mu\boldsymbol{R})\tilde{\boldsymbol{c}}_{k-1} \qquad k \geq 0 \tag{9.3.11}$$

which is a homogeneous difference equation. Using the principal-components transformation $\boldsymbol{R} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^{\mathrm{H}}$, we can write (9.3.11) as

$$\tilde{\boldsymbol{c}}'_k = (\boldsymbol{I} - 2\mu\Lambda)\tilde{\boldsymbol{c}}'_{k-1} \qquad k \geq 0 \tag{9.3.12}$$

where

$$\tilde{\boldsymbol{c}}'_k = \boldsymbol{Q}^{\mathrm{H}}\tilde{\boldsymbol{c}}_k \qquad k \geq 0 \tag{9.3.13}$$

is the transformed coefficient error vector. Since Λ is diagonal, (9.3.12) consists of a set of *M* decoupled first-order

difference equations

$$\tilde{c}'_{k,i} = (1-2\mu\lambda_i)\tilde{c}'_{k-1,i} \qquad i=1,\ 2,\ \cdots,\ M,\ k \geq 0 \tag{9.3.14}$$

with each describing a *natural mode* of the SDA. The solutions of (9.3.12) are given by

$$\tilde{c}'_{k,i} = (1-2\mu\lambda_i)^k \tilde{c}'_{0,i} \qquad k \geq 0 \tag{9.3.15}$$

If for all $1 \leq i \leq M$

$$-1 < 1-2\mu\lambda_i < 1 \tag{9.3.16}$$

or equivalently

$$0 < \mu < \frac{1}{\lambda_i} \tag{9.3.17}$$

then $\tilde{c}'_{k,i}, 1 \leq i \leq M$, tends to zero as $k \to \infty$. This implies that $\boldsymbol{c}_k$ converges *exponentially* to $\boldsymbol{c}_o$ as $k \to \infty$ because $\|\tilde{\boldsymbol{c}}'_k\| = \|\boldsymbol{Q}^T \tilde{\boldsymbol{c}}_k\| = \|\tilde{\boldsymbol{c}}_k\|$. If $\boldsymbol{R}$ is positive definite, its eigenvalues are positive and

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{9.3.18}$$

provides a necessary and sufficient condition for the convergence of SDA.

To investigate the transient behavior of the SDA as a function of k, we note that using (9.3.10), (9.3.11), and (9.3.14), we have

$$c_{k,i} = c_{o,i} + \sum_{i=1}^{M} q_{ik}\tilde{c}'_{0,i}(1-2\mu\lambda_i)^k \tag{9.3.19}$$

where $c_{o,i}$ are the optimum coefficients and q_{ik} the elements of the eigenvector matrix $\boldsymbol{Q}$. The MSE at step k is

$$P_k = P_o + \sum_{i=1}^{M} \lambda_i (1-2\mu\lambda_i)^{2k} \, |\tilde{c}'_{0,i}|^2 \tag{9.3.20}$$

and can be obtained by substituting (9.3.19) in (9.3.1). If μ satisfies (9.3.18), we have $\lim_{k\to\infty} P_k = P_o$ and the MSE converges exponentially to the optimum value. The curve obtained by plotting the MSE P_k as a function of the number of iterations k is known as the *learning curve*.

Rate of convergence

The rate (or speed) of convergence depends upon the algorithm and the nature of the performance surface. The most influential effect is inflicted by the condition number of the Hessian matrix that determines the shape of the contours of $P(\boldsymbol{c})$. When $P(\boldsymbol{c})$ is quadratic, it can be shown (Luenberger 1984) that

$$P(\boldsymbol{c}_k) \leq \left[\frac{\chi(\boldsymbol{R})-1}{\chi(\boldsymbol{R})+1}\right]^2 P(\boldsymbol{c}_{k-1}) \tag{9.3.21}$$

where $\chi(\boldsymbol{R}) = \lambda_{\max}/\lambda_{\min}$ is the condition number of $\boldsymbol{R}$. If we recall that the eigenvectors corresponding to $\lambda_{\min}$ and $\lambda_{\max}$ point to the directions of minimum and maximum curvature, respectively, we see that the convergence slows down as the contours become more eccentric (flattened). For circular contours, that is, when $\chi(\boldsymbol{R}) = 1$, the algorithm converges in one step. We stress that even if the $M-1$ eigenvalues of $\boldsymbol{R}$ are equal and the remaining one is far away, still the convergence of the SDA is very slow.

The rate of convergence can be characterized by using the *time constant* τ_i defined by

$$1-2\mu\lambda_i = \exp\left(-\frac{1}{\tau_i}\right) \simeq 1-\frac{1}{\tau_i} \tag{9.3.22}$$

which provides the time (or number of iterations) it takes for the ith mode $c_{k,i}$ of (9.3.19) to decay to $1/e$ of its initial value $c_{0,i}$. When $\mu \ll 1$, we obtain

$$\tau_i \simeq \frac{1}{2\mu\lambda_i} \tag{9.3.23}$$

In a similar fashion, the time constant $\tau_{i,\text{mse}}$ for the MSE P_k can be shown to be

$$\tau_{i,\text{mse}} \simeq \frac{1}{4\mu\lambda_i} \tag{9.3.24}$$

by using (9.3.20) and (9.3.22).

Thus, for all practical purposes, the time constant (for coefficient $\boldsymbol{c}_k$ or for MSE $\boldsymbol{P}_k$) of the SDA is $\tau \simeq 1/(\mu\lambda_{\min})$, which in conjunction with $\mu < 1/\lambda_{\max}$ results in $\tau > \lambda_{\max}/\lambda_{\min}$. Hence, *the larger the eigenvalue spread of the input correlation matrix* $\boldsymbol{R}$*, the longer it takes for the SDA to converge.*

In the following example, we illustrate above-discussed properties of the SDA by using it to compute the parameters of a second-order forward linear predictor.

EXAMPLE 9.3.1. Consider a signal generated by the second-order autoregressive AR(2) process

$$x(n) + a_1 x(n-1) + a_2 x(n-2) = \omega(n) \tag{9.3.25}$$

where $\omega(n) \sim \text{WGN}(0, \sigma_\omega^2)$. Parameters a_1 and a_2 are chosen so that the system (9.3.25) is minimum-phase. We want to design an adaptive filter that uses the samples $x(n-1)$ and $x(n-2)$ to predict the value $x(n)$ (desired response).

If we multiply (9.3.25) by $x(n-k)$, for $k = 0, 1, 2$, and take the mathematical expectation of both sides, we obtain a set of linear equations

$$r(0) + a_1 r(1) + a_2 r(2) = \sigma_\omega^2 \tag{9.3.26}$$

$$r(1) + a_1 r(0) + a_2 r(1) = 0 \tag{9.3.27}$$

$$r(2) + a_1 r(1) + a_2 r(0) = 0 \tag{9.3.28}$$

which can be used to express the autocorrelation of $x(n)$ in terms of model parameters a_1, a_2, and σ_ω^2. Indeed, solving (9.3.26) through (9.3.28), we obtain

$$\begin{aligned} r(0) &= \sigma_x^2 = \frac{1+a_2}{1-a_2} \frac{\sigma_\omega^2}{(1+a_2)^2 - a_1^2} \\ r(1) &= \frac{-a_1}{1+a_2} r(0) \\ r(2) &= \left(-a_2 + \frac{a_1^2}{1+a_2}\right) r(0) \end{aligned} \tag{9.3.29}$$

We choose $\sigma_x^2 = 1$, so that

$$\sigma_\omega^2 = \frac{(1-a_2)[(1+a_2)^2 - a_1^2]}{1+a_2} \sigma_x^2 \tag{9.3.30}$$

The coefficients of the optimum predictor

$$\hat{y}(n) = \hat{x}(n) = c_{o,1} x(n-1) + c_{o,2} x(n-2) \tag{9.3.31}$$

are given by (see Section 6.5)

$$r(0) c_{o,1} + r(1) c_{o,2} = r(1) \tag{9.3.32}$$

$$r(1) c_{o,1} + r(0) c_{o,2} = r(2) \tag{9.3.33}$$

with

$$P_o^f = r(0) + r(1) c_{o,1} + r(0) c_{o,2} \tag{9.3.34}$$

whose comparison with (9.3.26) through (9.3.28) shows that $c_{o,1} = -a_1$, $c_{o,2} = -a_2$, and $P_o^f = \sigma_\omega^2$, as expected.

The eigenvalues of the input correlation matrix

$$\boldsymbol{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \tag{9.3.35}$$

are

$$\lambda_{1,2} = \left(1 \mp \frac{a_1}{1+a_2}\right)\sigma_x^2 \tag{9.3.36}$$

from which the eigenvalue spread is

$$\chi(\boldsymbol{R}) = \frac{\lambda_1}{\lambda_2} = \frac{1-a_1+a_2}{1+a_1+a_2} \tag{9.3.37}$$

which, if $a_2 > 0$ and $a_1 < 0$, is larger than 1.

Now we perform MATLAB experiments with varying eigenvalue spread $\chi(\boldsymbol{R})$ and step-size parameter μ. In these experiments, we choose σ_ω^2 so that $\sigma_x^2 = 1$. The SDA is given by

$$\boldsymbol{c}_k \triangleq [c_{k,1} \;\; c_{k,2}]^{\mathrm{T}} = \boldsymbol{c}_{k-1} + 2\mu(\boldsymbol{d} - \boldsymbol{R}\boldsymbol{c}_{k-1})$$

where

$$\boldsymbol{d} = [r(1) \; r(2)]^{\mathrm{T}} \quad \text{and} \quad \boldsymbol{c}_0 = [0 \; 0]^{\mathrm{T}}$$

We choose two different sets of values for a_1 and a_2, one for a small and the other for a large eigenvalue spread. These values are shown in Table 9.2 along with the corresponding eigenvalue spread $\chi(\boldsymbol{R})$ and the MMSE σ_ω^2.

TABLE 9.2
Parameter values used in the SDA for the second-order forward prediction problem.

Eigenvalue spread	a_1	a_2	λ_1	λ_2	$\chi(\boldsymbol{R})$	σ_ω^2
Small	–0.1950	0.95	1.1	0.9	1.22	0.0965
Large	−1.5955	0.95	1.818	0.182	9.99	0.0322

Using each set of parameter values, the SDA is implemented starting with the null coefficient vector $\boldsymbol{c}_0$ with two values of step-size parameters. To describe the transient behavior of the algorithm, it is informative to plot the trajectory of $c_{k,1}$ versus $c_{k,2}$ as a function of the iteration index k along with the contours of the error surface $P(\boldsymbol{c}_k)$. The trajectory of $\boldsymbol{c}_k$ begins at the origin $\boldsymbol{c}_0$ = 0 and ends at the optimum value $\boldsymbol{c}_o = -[a_1 \;\; a_2]^{\mathrm{T}}$. This illustration of the transient behavior can also be obtained in the domain of the transformed error coefficients $\tilde{\boldsymbol{c}}_k'$. Using (9.3.15), we see these coefficients are given by

$$\tilde{\boldsymbol{c}}_k' = \begin{bmatrix} \tilde{c}_{k,1}' \\ \tilde{c}_{k,2}' \end{bmatrix} = \begin{bmatrix} (1-2\mu\lambda_1)^k \tilde{c}_{0,1}' \\ (1-2\mu\lambda_2)^k \tilde{c}_{0,2}' \end{bmatrix} \tag{9.3.38}$$

where $\tilde{\boldsymbol{c}}_0'$ from (9.3.10) and (9.3.13) is given by

$$\tilde{\boldsymbol{c}}_0' = \begin{bmatrix} \tilde{c}_{0,1}' \\ \tilde{c}_{0,1}' \end{bmatrix} = \boldsymbol{Q}^{\mathrm{T}}\tilde{\boldsymbol{c}}_0 = \boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{c}_0 - \boldsymbol{c}_o) = -\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{c}_o = \boldsymbol{Q}^{\mathrm{T}}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{9.3.39}$$

Thus the trajectory of $\tilde{\boldsymbol{c}}_k'$ begins at $\tilde{\boldsymbol{c}}_0'$ and ends at the origin $\tilde{\boldsymbol{c}}_k' = 0$. The contours of the MSE function in the transformed domain are given by $P_k - P_o$. From (9.3.20), these contours are given by

$$P_k - P_o^f = \sum_{i=1}^{2} \lambda_i (\tilde{\boldsymbol{c}}_k')^2 = \lambda_1 (\tilde{c}_{k,1}')^2 + \lambda_2 (\tilde{c}_{k,2}')^2 \tag{9.3.40}$$

Small eigenvalue spread and overdamped response. For this experiment, the parameter values were selected to obtain the eigenvalue spread approximately equal to 1 [$\chi(\boldsymbol{R}) = 1.22$]. The step size selected was $\mu = 0.15$, which is less than $1/\lambda_{\max} = 1/1.1 = 0.9$ for convergence. For this value of μ, the transient response is overdamped. Figure 9.14 shows four graphs indicating the behavior of the algorithm. In the graph (*a*), the trajectory of $\tilde{\boldsymbol{c}}_k'$ is shown for $0 \le k \le 15$ along with the corresponding loci $\tilde{\boldsymbol{c}}_k'$ for a fixed value of $P_k - P_o$. The first two loci for $k = 0$ and 1 are numbered to show the direction of the trajectory. Graph (*b*) shows the corresponding trajectory and the contours for $\boldsymbol{c}_k$. Graph (*c*) shows plots of $c_{k,1}$ and $c_{k,2}$ as a function of iteration step k, while graph (*d*) shows a similar learning curve for the MSE P_k. Several observations can be made about these plots. The contours of constant $\tilde{\boldsymbol{c}}_k'$ are almost circular since the spread is approximately 1, while those of $\boldsymbol{c}_k$

are somewhat elliptical, which is to be expected. The trajectories of $\tilde{c}'_k$ and c_k as a function of k are normal to the contours. The coefficients converge to their optimum values in a monotonic fashion, which confirms the overdamped nature of the response. Also this convergence is rapid, in about 15 steps, which is to be expected for a small eigenvalue spread.

Large eigenvalue spread and overdamped response. For this experiment, the parameter values were selected so that the eigenvalue spread was approximately equal to 10 [$\chi(\boldsymbol{R}) = 9.99$]. The step size was again selected as $\mu = 0.15$. Figure 9.15 shows the performance plots for this experiment, which are similar to those of Figure 9.14. The observations are also similar except for those due to the larger spread. First, the contours, even in the transformed domain, are elliptical; second, the convergence is slow, requiring about 60 steps in the algorithm. The transient response is once again overdamped.

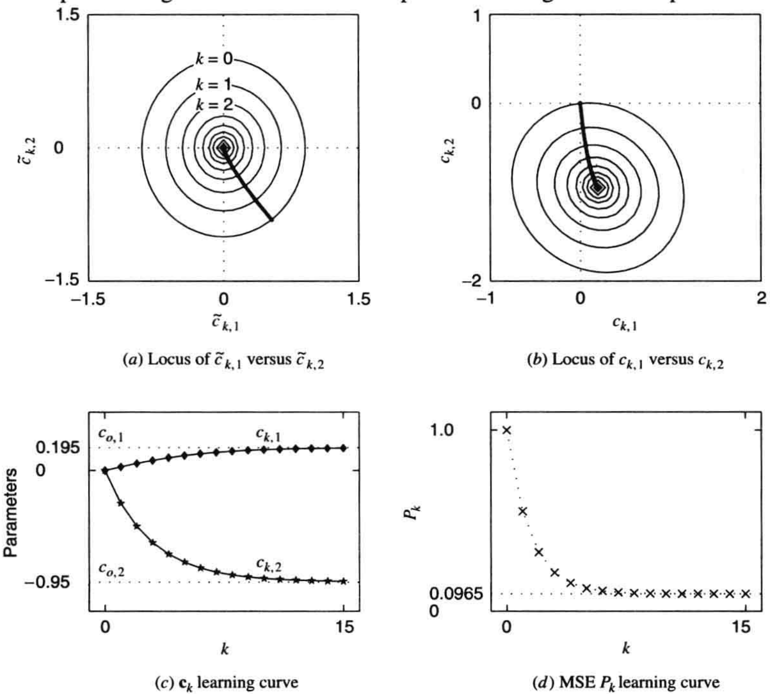


FIGURE 9.14

Performance curves for the steepest-descent algorithm used in the linear prediction problem with step-size parameter $\mu = 0.15$ and eigenvalue spread $\chi(\boldsymbol{R}) = 1.22$.

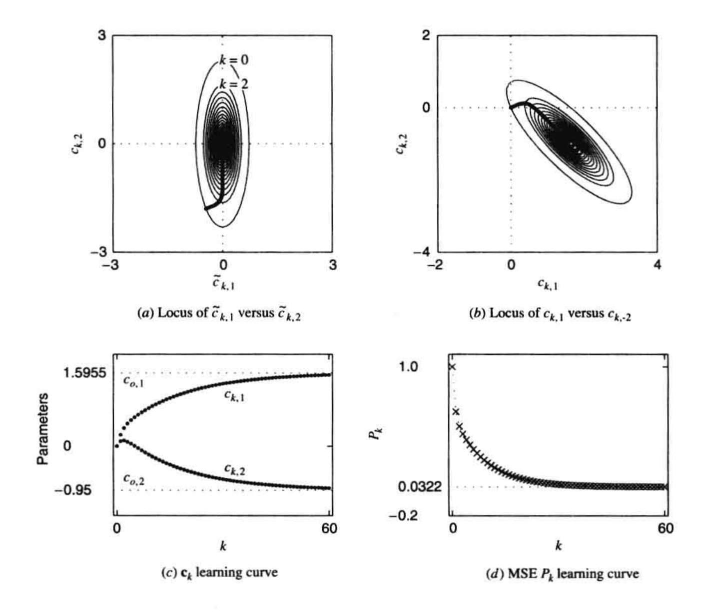


FIGURE 9.15

Performance curves for the steepest-descent algorithm used in the linear prediction problem with step-size parameter $\mu = 0.15$ and eigenvalue spread $\chi(\boldsymbol{R}) = 10$.

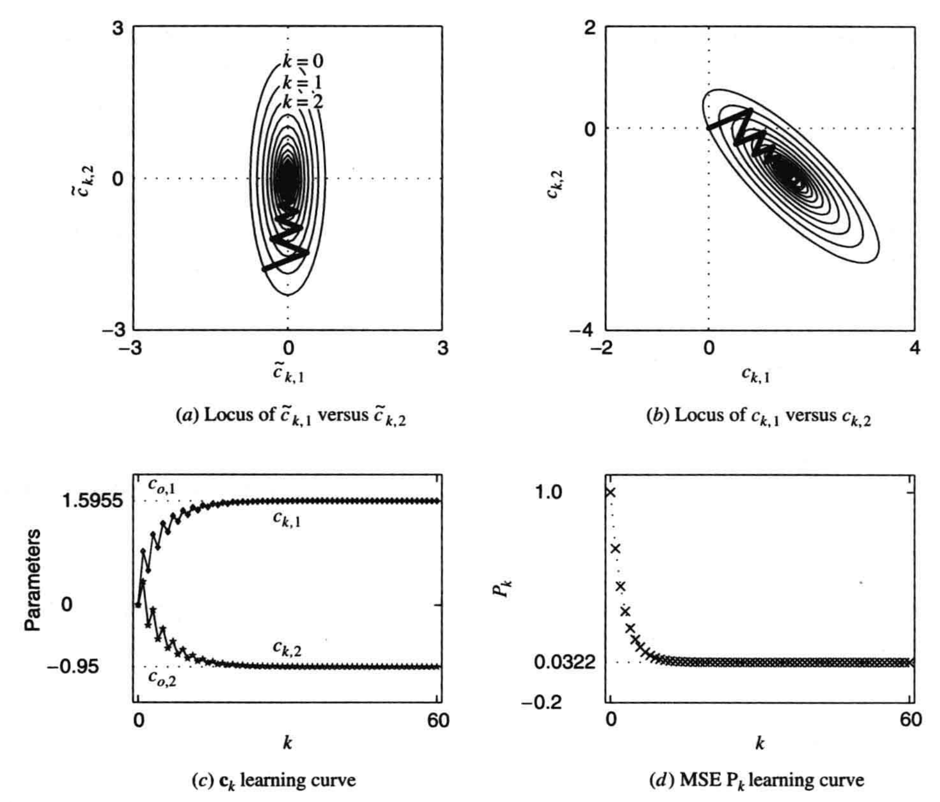


FIGURE 9.16
Performance curves for the steepest-descent algorithm used in the linear prediction problem with eigenvalue spread $\chi(\boldsymbol{R}) = 10$ and varying step-size parameters $\mu = 0.15$ and $\mu = 0.5$.

Note that the coefficients converge in an oscillatory fashion; however, the convergence is fairly rapid compared to that of the overdamped case. Thus the selection of the step size is an important design issue.

Newton's type of algorithms

Another family of algorithms with a faster rate of convergence includes Newton's method and its modifications. The basic idea of Newton's method is to achieve convergence in one step when $P(\mathbf{c})$ is quadratic. Thus, if $\mathbf{c}_k$ is to be the minimum of $P(\boldsymbol{c})$, the gradient $\nabla P(\boldsymbol{c}_k)$ of $P(\boldsymbol{c})$ evaluated at $\boldsymbol{c}_k$ (9.2.19) should be zero. From (9.2.19), we can write

$$\nabla P(\boldsymbol{c}_k) = \nabla P(\boldsymbol{c}_{k-1}) + \nabla^2 P(\boldsymbol{c}_{k-1}) \Delta \boldsymbol{c}_k = 0 \tag{9.3.41}$$

Thus $\nabla P(\boldsymbol{c}_k) = 0$ leads to the step increment

$$\Delta \boldsymbol{c}_k = -[\nabla^2 P(\boldsymbol{c}_{k-1})]^{-1} \nabla P(\boldsymbol{c}_{k-1}) \tag{9.3.42}$$

and hence the adaptive algorithm is given by

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} - \mu [\nabla^2 P(\boldsymbol{c}_{k-1})]^{-1} \nabla P(\boldsymbol{c}_{k-1}) \tag{9.3.43}$$

where $\mu > 0$ is the step size. For quadratic error surfaces, from (9.3.4) and (9.3.5), we obtain with $\mu = 1$

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} - [\nabla^2 P(\boldsymbol{c}_{k-1})]^{-1} \nabla P(\boldsymbol{c}_{k-1}) = \boldsymbol{c}_{k-1} - (\boldsymbol{c}_{k-1} - \boldsymbol{R}^{-1}\boldsymbol{d}) = \boldsymbol{c}_o \tag{9.3.44}$$

which shows that indeed the algorithm converges in one step.

For the quadratic case, since $\nabla^2 P(\boldsymbol{c}_{k-1}) = 2\boldsymbol{R}$ from (9.3.1), we can express Newton's algorithm as

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} - \mu \boldsymbol{R}^{-1} \nabla P(\boldsymbol{c}_{k-1}) \tag{9.3.45}$$

where μ is the step size that regulates the convergence rate. Other modified Newton methods replace the Hessian matrix $\nabla^2 P(c_{k-1})$ with another matrix, which is guaranteed to be positive definite and, in some way, close to the Hessian. These Newton-type algorithms generally provide faster convergence. However, in practice, the inversion of **R** is numerically intensive and can lead to a numerically unstable solution if special care is not taken. Therefore, the SDA is more popular in adaptive filtering applications.

When the function $P(c)$ is nonquadratic, it is approximated locally by a quadratic function that is minimized exactly. However, the step obtained in (9.3.42) does not lead to the minimum of $P(c)$, and the iteration should be repeated several times. A more detailed treatment of linear and nonlinear optimization techniques can be found in Scales (1985) and in Luenberger (1984).

9.4 Least-Mean-Square Adaptive Filters

In this section, we derive, analyze the performance, and present some practical applications of the *least-mean-square* (*LMS*) adaptive algorithm. The LMS algorithm, introduced by Widrow and Hoff (1960), is widely used in practice due to its simplicity, computational efficiency, and good performance under a variety of operating conditions.

9.4.1 Derivation

We first present two approaches to the derivation of the LMS algorithm that will help the reader to understand its operation. The first approach uses approximation to the gradient function while the second approach uses geometric arguments.

Optimization approach. The SDA uses the second-order moments $\boldsymbol{R}$ and $\boldsymbol{d}$ to iteratively compute the optimum filter $\boldsymbol{c}_o = \boldsymbol{R}^{-1}\boldsymbol{d}$, starting with an initial guess, usually $\mathbf{c}_0 = 0$, and then obtaining better approximations by taking steps in the direction of the negative gradient, that is,

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + \mu[-\nabla P(\boldsymbol{c}_{k-1})] \tag{9.4.1}$$

where

$$\nabla P(\boldsymbol{c}_{k-1}) = 2(\boldsymbol{R}\boldsymbol{c}_{k-1} - \boldsymbol{d}) \tag{9.4.2}$$

is the gradient of the performance function (9.3.1). In practice, where only the input $\{x(j)\}_0^n$ and the desired response $\{y(j)\}_0^n$ are known, we can only compute an estimate of the "true" or exact gradient (9.4.2) using the available data. To develop an adaptive algorithm from (9.4.1), we take the following steps: (1) replace the iteration subscript k by the time index n; and (2) replace $\boldsymbol{R}$ and $\boldsymbol{d}$ by their instantaneous estimates $\boldsymbol{x}(n)\boldsymbol{x}^H(n)$ and $\boldsymbol{x}(n)y^*(n)$, respectively. The instantaneous estimate of the gradient (9.4.2) becomes

$$\nabla P(\boldsymbol{c}_{k-1}) = 2\boldsymbol{R}\boldsymbol{c}_{k-1} - 2\boldsymbol{d} \simeq 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{c}(n-1) - 2\boldsymbol{x}(n)y^*(n) = -2\boldsymbol{x}(n)e^*(n) \tag{9.4.3}$$

where

$$e(n) = y(n) - \boldsymbol{c}^H(n-1)\boldsymbol{x}(n) \tag{9.4.4}$$

is the a priori filtering error. The estimate (9.4.3) also can be obtained by starting with the approximation $P(\boldsymbol{c}) \simeq |e(n)|^2$ and taking its gradient. The coefficient adaptation algorithm is

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + 2\mu\boldsymbol{x}(n)e^*(n) \tag{9.4.5}$$

which is obtained by substituting (9.4.3) and (9.4.4) in (9.4.1). The step-size parameter 2μ is also known as the *adaptation gain*.

The LMS algorithm, specified by (9.4.5) and (9.4.4), has both important similarities to and important differences from the SDA (9.3.7). The SDA contains deterministic quantities while the LMS operates on random quantities. The SDA is *not* an adaptive algorithm because it only depends on the second-order moments $\boldsymbol{R}$ and $\boldsymbol{d}$ and not on the SOE $\{\boldsymbol{x}(n,\zeta), y(n,\zeta)\}$. Also, the iteration index k has nothing to do with time. Simply stated, the SDA provides an iterative solution to the linear system $\boldsymbol{Rc} = \boldsymbol{d}$.

Geometric approach. Suppose that an adaptive filter operates in a stationary signal environment seeking the optimum filter $\boldsymbol{c}_o$. At time n the filter has access to input vector $\boldsymbol{x}(n)$, the desired response $y(n)$, and the previous or old coefficient estimate $\boldsymbol{c}(n-1)$. Its goal is to use this information to determine a new estimate $\boldsymbol{c}(n)$ that is closer to the optimum vector $\mathbf{c}_o$ or equivalently to choose $\boldsymbol{c}(n)$ so that $\|\tilde{c}(n)\| < \|\tilde{c}(n-1)\|$, where

$\tilde{c}(n) = c(n) - c_o$ is the coefficient error vector given by (9.2.24). Eventually, we want $\|\tilde{c}(n)\|$ to become negligible as $n \to \infty$.

The vector $\tilde{c}(n-1)$ can be decomposed into two orthogonal components

$$\tilde{c}(n-1) = \tilde{c}_x(n-1) + \tilde{c}_x^{\perp}(n-1) \tag{9.4.6}$$

one parallel and one orthogonal to the input vector $x(n)$, as shown in Figure 9.17(a). The response of the *error filter* $\tilde{c}(n-1)$ to the input $x(n)$ is

$$\tilde{y}(n) = \tilde{c}^H(n-1)x(n) = \tilde{c}_x^H(n-1)x(n) \tag{9.4.7}$$

which implies that

$$\tilde{c}_x(n-1) = \frac{\tilde{y}^*(n)}{\|x(n)\|^2} x(n) \tag{9.4.8}$$

which can be verified by direct substitution in (9.4.7). Note that $x(n)/\|x(n)\|$ is a unit vector along the direction of $\mathbf{x}(n)$.

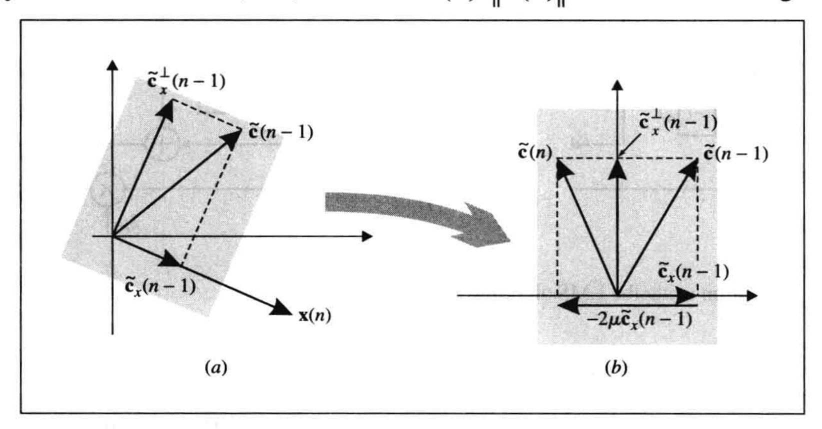


FIGURE 9.17
The geometric approach for the derivation of the LMS algorithm.

If we only know $x(n)$ and $\tilde{y}(n)$, the best strategy to decrease $\tilde{c}(n)$ is to choose $\tilde{c}(n) = \tilde{c}_x^{\perp}(n-1)$, or equivalently subtract $\tilde{c}_x(n-1)$ from $\tilde{c}(n-1)$. From Figure 9.17(a) note that as long as $\tilde{c}_x(n-1) \neq 0$, $\|\tilde{c}(n)\| = \|\tilde{c}_x^{\perp}(n-1)\| < \|\tilde{c}_x(n-1)\|$. This suggests the following adaptation algorithm

$$\tilde{c}(n) = \tilde{c}(n-1) - \tilde{\mu}\frac{\tilde{y}^*(n)}{\|x(n)\|^2} x(n) \tag{9.4.9}$$

which guarantees that $\|\tilde{c}(n)\| < \|\tilde{c}(n-1)\|$ as long as $0 < \tilde{\mu} < 2$ and $\tilde{y}(n) \neq 0$, as shown in Figure 9.17 (b). The best choice clearly is $\tilde{\mu} = 1$.

Unfortunately, the signal $\tilde{y}(n)$ is not available, and we have to replace it with some reasonable approximation. From (9.2.18) and (9.2.10) we obtain

$$\begin{aligned} \tilde{e}(n) &\triangleq e(n) - e_o(n) = y(n) - \hat{y}(n) - y(n) + \hat{y}_o(n) = \hat{y}_o(n) - \hat{y}(n) \\ &= [\mathbf{c}_o^H - \mathbf{c}^H(n-1)]\mathbf{x}(n) = -\tilde{c}^H(n-1)x(n) = -\tilde{y}(n) \end{aligned} \tag{9.4.10}$$

where we have used (9.4.7). Using the approximation

$$\tilde{e}(n) = e(n) - e_o(n) \simeq e(n)$$

we combine it with (9.4.10) to get

$$c(n) = c(n-1) + \tilde{\mu}\frac{e^*(n)}{\|x(n)\|^2} x(n) \tag{9.4.11}$$

which is known as the *normalized LMS algorithm*. Note that the effective step size $\tilde{\mu}/\|\mathbf{x}(n)\|^2$ is time-varying. The LMS algorithm in (9.4.5) follows if we set $\|\boldsymbol{x}(n)\|=1$ and choose $\tilde{\mu}=2\mu$.

LMS algorithm. The LMS algorithm can be summarized as

$$\begin{aligned} \hat{y}(n) &= \boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n) && \text{filtering} \\ e(n) &= y(n)-\hat{y}(n) && \text{error formation} \\ \boldsymbol{c}(n) &= \boldsymbol{c}(n-1)+2\mu\boldsymbol{x}(n)e^{*}(n) && \text{coefficient updating} \end{aligned} \tag{9.4.12}$$

where μ is adaptation step size. The algorithm requires $2M+1$ complex multiplications and $2M$ complex additions. Figure 9.18 shows an implementation of an FIR adaptive filter using the LMS algorithm, which is implemented in MATLAB using the function `[yhat,c]= firlms(x,y,M,mu)`. The a posteriori form of the LMS algorithm is developed in Problem 9.9.

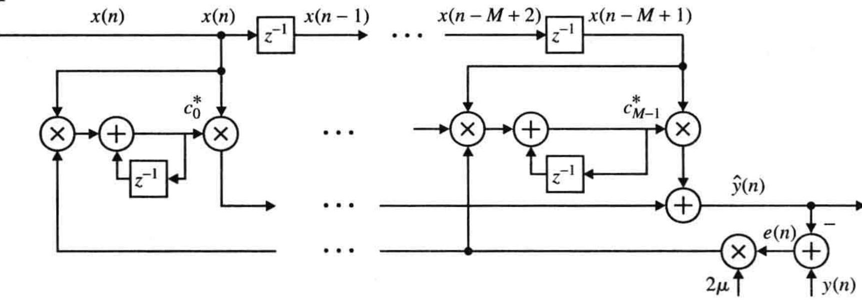


FIGURE 9.18
An FIR adaptive filter realization using the LMS algorithm.

9.4.2 *Adaptation in a Stationary SOE*

In the sequel, we study the stability and steady-state performance of the LMS algorithm in a stationary SOE; that is, we assume that the input and the desired response processes are jointly stationary. In theory, the goal of the LMS adaptive filter is to identify the optimum filter $\boldsymbol{c}_o=\boldsymbol{R}^{-1}\boldsymbol{d}$ from observations of the input $\boldsymbol{x}(n)$ and the desired response

$$y(n)=\boldsymbol{c}_o^{\mathrm{H}}\boldsymbol{x}(n)+e_o(n) \tag{9.4.13}$$

The optimum error $e_o(n)$ is orthogonal to the vector $\boldsymbol{x}(n)$; that is, $E\{\boldsymbol{x}(n)e^{*}(n)\}=\mathbf{0}$ and acts as measurement or output noise, as shown in Figure 9.19.

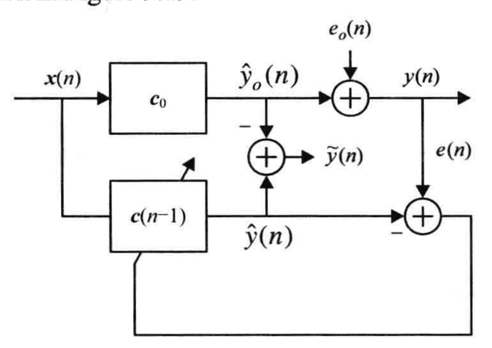


FIGURE 9.19
LMS algorithm in a stationary SOE.

The first step in the statistical analysis of the LMS algorithm is to determine a difference equation for the coefficient error vector $\tilde{\mathbf{c}}(n)$. To this end, we subtract $\mathbf{c}_o$ from both sides of (9.4.5), to obtain

$$\tilde{\boldsymbol{c}}(n)=\tilde{\boldsymbol{c}}(n-1)+2\mu\boldsymbol{x}(n)e^{*}(n) \tag{9.4.14}$$

which expresses the LMS algorithm in terms of the coefficient error vector. We next use (9.4.12) and (9.4.13) in (9.4.14) to eliminate $e(n)$ by expressing it in terms of $\tilde{\mathbf{c}}(n-1)$ and $e_o(n)$. The result is

$$\tilde{\boldsymbol{c}}(n) = [\boldsymbol{I} - 2\mu \boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]\tilde{\boldsymbol{c}}(n-1) + 2\mu \boldsymbol{x}(n)e_o^*(n) \tag{9.4.15}$$

which is a time-varying forced or nonhomogeneous stochastic difference equation. The irreducible error $e_o(n)$ accounts for measurement noise, modeling errors, unmodeled dynamics, quantization effects, and other disturbances. The presence of $e_o(n)$ prevents convergence because it forces $\tilde{\boldsymbol{c}}(n)$ to fluctuate around zero. Therefore, the important issue is the BIBO stability of the system (9.4.15). From (9.2.28), we see that $\|\tilde{\boldsymbol{c}}(n)\|$ is bounded in mean square if we can show that $E\{\tilde{\boldsymbol{c}}(n)\} \to 0$ as $n \to \infty$ and $var\{\tilde{c}_k(n)\}$ is bounded for all n. To this end, we develop difference equations for the mean value $E\{\tilde{\boldsymbol{c}}(n)\}$ and the correlation matrix

$$\boldsymbol{\Phi}(n) \triangleq E\{\tilde{\boldsymbol{c}}(n)\tilde{\boldsymbol{c}}^{\mathrm{H}}(n)\} \tag{9.4.16}$$

of the coefficient error vector $\tilde{\boldsymbol{c}}(n)$. As we shall see, the MSD and the EMSE can be expressed in terms of matrices $\boldsymbol{\Phi}(n)$ and $\boldsymbol{R}$. The time evolution of these quantities provides sufficient information to evaluate the stability and steady-state performance of the LMS algorithm.

Convergence of the mean coefficient vector

If we take the expectation of (9.4.15), we have

$$E\{\tilde{\boldsymbol{c}}(n)\} = E\{\tilde{\boldsymbol{c}}(n-1)\} - 2\mu E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}(n-1)\} \tag{9.4.17}$$

because $E\{\boldsymbol{x}(n)e_o^*(n)\} = \boldsymbol{0}$ owing to the orthogonality principle. The computation of the second term in (9.4.17) requires the correlation between the input signal and the coefficient error vector.

If we assume that $\boldsymbol{x}(n)$ and $\tilde{\boldsymbol{c}}(n-1)$ are statistically independent, (9.4.17) simplifies to

$$E\{\tilde{\boldsymbol{c}}(n)\} = (\boldsymbol{I} - 2\mu\boldsymbol{R})E\{\tilde{\boldsymbol{c}}(n-1)\} \tag{9.4.18}$$

which has the same form as (9.3.11) for the SDA. Therefore, $\tilde{\boldsymbol{c}}(n)$ converges in the MS sense, that is, $\lim_{n\to\infty} E\{\tilde{\boldsymbol{c}}(n)\} = \boldsymbol{0}$, if the eigenvalues of the system matrix $(\boldsymbol{I} - 2\mu\boldsymbol{R})$ are less than 1. Hence, if $\boldsymbol{R}$ is positive definite and $\lambda_{\max}$ is its maximum eigenvalue, the condition

$$0 < 2\mu < \frac{1}{\lambda_{\max}} \tag{9.4.19}$$

ensures that the LMS algorithm converges in the MS sense [see the discussion following (9.2.27)].

Independence assumption. The independence assumption between $\boldsymbol{x}(n)$ and $\tilde{\boldsymbol{c}}(n-1)$ was critical to the derivation of (9.4.18). To simplify the analysis, we make the following *independence assumptions* (Gardner 1984):

A1 The sequence of input data vectors $\boldsymbol{x}(n)$ is independently and identically distributed with zero mean and correlation matrix $\boldsymbol{R}$.

A2 The sequences $\boldsymbol{x}(n)$ and $e_o(n)$ are independent for all n.

From (9.4.15), we see that $\tilde{\boldsymbol{c}}(n-1)$ depends on $\tilde{\boldsymbol{c}}(0)$, $\{\boldsymbol{x}(k)\}_0^{n-1}$, and $\{e_o(k)\}_0^{n-1}$. Since the sequence $\boldsymbol{x}(n)$ is IID and the quantities $\boldsymbol{x}(n)$ and $e_o(n)$ are independent, we conclude that $\boldsymbol{x}(n)$, $e_o(n)$, and $\tilde{\boldsymbol{c}}(n-1)$ are mutually independent. This result will be used several times to simplify the analysis of the LMS algorithm.

The independence assumption A1, first introduced in Widrow et al. (1976) and in Mazo (1979), ignores the statistical dependence among successive input data vectors; however, it preserves sufficient statistical information about the adaptation process to lead to useful design guidelines. Clearly, for FIR filtering applications, the independence assumption is violated because two successive input data vectors $\boldsymbol{x}(n)$ and $\boldsymbol{x}(n+1)$ have $M-1$ common elements (shift-invariance property).

Evolution of the coefficient error correlation matrix

The MSD can be expressed in terms of the trace of the correlation matrix[①] $\Phi(n)$, that is,

$$D(n) = \mathrm{tr}[\Phi(n)] \tag{9.4.20}$$

which can be easily seen by using (9.2.29) and the definition of trace. If we postmultiply both sides of (9.4.15) by their respective Hermitian transposes and take the mathematical expectation, we obtain

$$\begin{aligned}\Phi(n) &= E\{\tilde{c}(n)\tilde{c}^{\mathrm{H}}(n)\} \\ &= E\{[\boldsymbol{I} - 2\mu\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]\tilde{c}(n-1)\tilde{c}^{\mathrm{H}}(n-1)[\boldsymbol{I} - 2\mu\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]^{\mathrm{H}}\} \\ &\quad + 2\mu E\{[\boldsymbol{I} - 2\mu\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]\tilde{c}(n-1)e_o(n)\boldsymbol{x}^{\mathrm{H}}(n)\} \\ &\quad + 2\mu E\{\boldsymbol{x}(n)e_o^*(n)\tilde{c}^{\mathrm{H}}(n-1)[\boldsymbol{I} - 2\mu\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]^{\mathrm{H}}\} \\ &\quad + 4\mu^2 E\{\boldsymbol{x}(n)e_o^*(n)e_o(n)\boldsymbol{x}^{\mathrm{H}}(n)\}\end{aligned} \tag{9.4.21}$$

From the independence assumptions, $e_o(n)$ is independent with $\tilde{c}(n-1)$ and $\boldsymbol{x}(n)$. Therefore, the second and third terms in (9.4.21) vanish, and the fourth term is equal to $4\mu^2 P_o\mathbf{R}$. If we expand the first term, we obtain

$$\Phi(n) = \Phi(n-1) - 2\mu[\boldsymbol{R}\Phi(n-1) + \Phi(n-1)\boldsymbol{R}] + 4\mu^2\boldsymbol{A} + 4\mu^2 P_o\boldsymbol{R} \tag{9.4.22}$$

where

$$A \triangleq E\{x(n)x^{\mathrm{H}}(n)\tilde{c}(n-1)\tilde{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)\} \tag{9.4.23}$$

and the terms $\boldsymbol{R}\,\boldsymbol{\Phi}(n-1)$ and $\boldsymbol{\Phi}(n-1)\boldsymbol{R}$ have been computed by using the mutual independence of $\boldsymbol{x}(n),\ \tilde{c}(n-1)$, and $e_o(n)$.

The computation of matrix **A** can be simplified if we make additional assumptions about the statistical properties of $\boldsymbol{x}(n)$. As shown in Gardner (1984), development of a recursive relation for the elements of $\boldsymbol{\Phi}(n)$ using only the indenpendent assumptions requires the products with and the inversion of a $M^2 \times M^2$ matrix, where *M* is the size of $\boldsymbol{x}(n)$.

The evaluation of this term when $\boldsymbol{x}(n) \sim$IID, an assumption that is more appropriate for data transmission applications, is discussed in Gardner (1984). The computation for $\boldsymbol{x}(n)$ being a *spherically invariant random process* (*SIRP*) is discussed in Rupp (1993). SIRP models, which include the Gaussian distribution as a special case, provide a good characterization of speech signals. However, independently of the assumption used, the basic conclusions remain the same.

Assuming that $\boldsymbol{x}(n)$ is normally distributed, that is, $\boldsymbol{x}(n) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$, a significant amount of simplification can be obtained. Indeed, in this case we can use the moment factorization property for normal random variables to express fourth-order moments in terms of second-order moments (Papoulis 1991). If z_1, z_2, z_3, and z_4 are complex-valued, zero-mean, and jointly distributed normal random variables, then

$$E\{z_1 z_2^* z_3 z_4^*\} = E\{z_1 z_2^*\}E\{z_3 z_4^*\} + E\{z_1 z_4^*\}E\{z_2^* z_3\} \tag{9.4.24}$$

or if they are real-valued, then

$$E\{z_1 z_2 z_3 z_4\} = E\{z_1 z_2\}E\{z_3 z_4\} + E\{z_1 z_3\}E\{z_2 z_4\} + E\{z_1 z_4\}E\{z_2 z_3\} \tag{9.4.25}$$

Using direct substitution of (9.4.24) or (9.4.25) in (9.4.23), we can show that

$$\boldsymbol{A} = \begin{cases} \boldsymbol{R\Phi}(n-1)\boldsymbol{R} + \boldsymbol{R}\,\mathrm{tr}[\boldsymbol{R\Phi}(n-1)] & \text{complex case} \\ 2\boldsymbol{R\Phi}(n-1)\boldsymbol{R} + \boldsymbol{R}\,\mathrm{tr}[\boldsymbol{R\Phi}(n-1)] & \text{real case} \end{cases} \tag{9.4.26}$$

Finally, substituting (9.4.26) in (9.4.22), we obtain a difference equation for $\boldsymbol{\Phi}(n)$. This is summarized in the following property:

[①] Note that when (10.4.19) holds, $\lim_{n\to\infty} E\{\tilde{\mathbf{c}}(n)\} = 0$, and therefore $\Phi(n)$ provides asymptotically the covariance of $\tilde{\mathbf{c}}(n)$.

PROPERTY 9.4.1. Using the independence assumptions A1 and A2, and the normal distribution assumption of $\mathbf{x}(n)$, the correlation matrix of the coefficient error vector $\tilde{\mathbf{c}}(n)$ satisfies the difference equation

$$\begin{aligned}\boldsymbol{\Phi}(n) &= \boldsymbol{\Phi}(n-1) - 2\mu[\mathbf{R}\boldsymbol{\Phi}(n-1) + \boldsymbol{\Phi}(n-1)\mathbf{R}] \\ &\quad + 4\mu^2 \mathbf{R}\boldsymbol{\Phi}(n-1)\mathbf{R} + 4\mu^2 \mathbf{R}\,\mathrm{tr}[\mathbf{R}\boldsymbol{\Phi}(n-1)] + 4\mu^2 P_o \mathbf{R}\end{aligned} \tag{9.4.27}$$

in the complex case and

$$\begin{aligned}\boldsymbol{\Phi}(n) &= \boldsymbol{\Phi}(n-1) - 2\mu[\mathbf{R}\boldsymbol{\Phi}(n-1) + \boldsymbol{\Phi}(n-1)\mathbf{R}] \\ &\quad + 8\mu^2 \mathbf{R}\boldsymbol{\Phi}(n-1)\mathbf{R} + 4\mu^2 \mathbf{R}\,\mathrm{tr}[\mathbf{R}\boldsymbol{\Phi}(n-1)] + 4\mu^2 P_o \mathbf{R}\end{aligned} \tag{9.4.28}$$

in the real case. Both relations are matrix difference equations driven by the constant term $4\mu^2 P_o \mathbf{R}$.

The presence of the term $4\mu^2 P_o \mathbf{R}$ in (9.4.27) or (9.4.28) implies that $\boldsymbol{\Phi}(n)$ will never become zero, and as a result the coefficients of the LMS adaptive filter will always fluctuate about their optimum settings, which prevents convergence. It has been shown (Bucklew et al. 1993) that asymptotically $\tilde{c}(n)$ follows a zero-mean normal distribution. The amount of fluctuation is measured by matrix $\boldsymbol{\Phi}(n)$. In contrast, the absence of a driving term in (9.4.18) allows the convergence of $E\{\mathbf{c}(n)\}$ to the optimum vector $\mathbf{c}_o$.

Since there are two distinct forms for the difference equation of $\boldsymbol{\Phi}(n)$, we will consider the *real* case (9.4.28) for further discussion. Similar analysis can be done for the complex case (9.4.27), which is undertaken in Problem 9.11. To further simplify the analysis, we transform $\boldsymbol{\Phi}(n)$ to the principal coordinate space of $\mathbf{R}$ using the spectral decomposition

$$\mathbf{Q}^{\mathrm{T}}\mathbf{R}\mathbf{Q} = \boldsymbol{\Lambda}$$

by defining the matrix

$$\boldsymbol{\Theta}(n) \triangleq \mathbf{Q}^{\mathrm{T}}\boldsymbol{\Phi}(n)\mathbf{Q} \tag{9.4.29}$$

which is symmetric and positive definite [when $\boldsymbol{\Phi}(n)$ is positive definite].

If we pre- and postmultiply (9.4.28) by $\mathbf{Q}^{\mathrm{T}}$ and $\mathbf{Q}$ and use $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}$, we obtain

$$\begin{aligned}\boldsymbol{\Theta}(n) &= \boldsymbol{\Theta}(n-1) - 2\mu[\boldsymbol{\Lambda}\boldsymbol{\Theta}(n-1) + \boldsymbol{\Theta}(n-1)\boldsymbol{\Lambda}] \\ &\quad + 8\mu^2 \boldsymbol{\Lambda}\boldsymbol{\Theta}(n-1)\boldsymbol{\Lambda} + 4\mu^2 \boldsymbol{\Lambda} tr[\boldsymbol{\Lambda}\boldsymbol{\Theta}(n-1)] + 4\mu^2 P_o \boldsymbol{\Lambda}\end{aligned} \tag{9.4.30}$$

which is easier to work with because of the diagonal nature of Λ. For any symmetric and positive definite matrix $\boldsymbol{\Theta}$, we have $|\theta_{ij}(n)|^2 \le \theta_{ii}\theta_{jj}$. Hence, the convergence of the diagonal elements ensures the convergence of the off-diagonal elements. This observation and (9.4.30) suggest that to analyze the LMS algorithm, we should extract from (9.4.30) the equations for the diagonal elements

$$\theta(n) \triangleq [\theta_1(n)\ \theta_2(n)\ \cdots\ \theta_M(n)]^{\mathrm{T}} \tag{9.4.31}$$

of $\Theta(n)$ and form a difference equation for the vector $\theta(n)$. Indeed, we can easily show that

$$\theta(n) = \mathbf{B}\theta(n-1) + 4\mu^2 P_o \lambda \tag{9.4.32}$$

where

$$\mathbf{B} \triangleq \boldsymbol{\Lambda}(\rho) + 4\mu^2 \lambda\lambda^{\mathrm{T}} \tag{9.4.33}$$

$$\boldsymbol{\lambda} \triangleq [\lambda_1\ \lambda_2\ \cdots\ \lambda_M]^{\mathrm{T}} \tag{9.4.34}$$

$$\boldsymbol{\Lambda}(\rho) \triangleq \mathrm{diag}\{\rho_1, \rho_2, \cdots, \rho_M\} \tag{9.4.35}$$

$$\rho_k = 1 - 4\mu\lambda_k + 8\mu^2\lambda_k^2 = (1 - 2\mu\lambda_k)^2 + 4\mu^2\lambda_k^2 > 0 \qquad 1 \le k \le M \tag{9.4.36}$$

and λ_k are the eigenvalues of $\mathbf{R}$. The solution of the vector difference equation (9.4.32) is

$$\boldsymbol{\theta}(n) = \mathbf{B}^n \boldsymbol{\theta}(0) + 4\mu^2 P_o \sum_{j=0}^{n-1} \mathbf{B}^j \boldsymbol{\lambda} \tag{9.4.37}$$

and can be easily found by recursion.

The stability of the linear system (9.4.32) is determined by the eigenvalues of the symmetric matrix $\mathbf{B}$. Using

(9.4.3) and (9.4.35), for an arbitrary vector **z**, we obtain

$$z^{\mathrm{T}}Bz = z^{\mathrm{T}}\Lambda(\rho)z + 4\mu^2(\lambda^{\mathrm{T}}z)^2 = \sum_{k=1}^{M}\rho_k z_k^2 + 4\mu^2(\lambda^{\mathrm{T}}z)^2 \tag{9.4.38}$$

where we have used (9.4.36). Hence (9.4.38), for $z \neq \mathbf{0}$, implies that $z^T Bz > 0$, that is, the matrix $\boldsymbol{B}$ is positive definite. Since matrix $\boldsymbol{B}$ is symmetric and positive definite, its eigenvalues $\lambda_k(\boldsymbol{B})$ are real and positive. The system (9.4.37) will be BIBO stable if and only if

$$0 < \lambda_k(\boldsymbol{B}) < 1 \qquad 1 \le k \le M \tag{9.4.39}$$

To find the range of μ that ensures (9.4.39), we use the Gerschgorin circles theorem (Noble and Daniel 1988), which states that *each eigenvalue of an* $M \times M$ *matrix* ***B*** *lies in at least one of the disks with center at the diagonal element* b_{kk} *and radius equal to the sum of absolute values* $|b_{kj}|, j \neq k$, *of the remaining elements of the row.* Since the elements of $\boldsymbol{B}$ are positive, we can easily see that

$$\lambda_k(\boldsymbol{B}) - b_{kk} < \sum_{\substack{j=1\\ j\neq k}}^{M} b_{kj} \quad \text{or} \quad \lambda_k(\boldsymbol{B}) < \rho_k + 4\mu^2\lambda_k\sum_{j=1}^{M}\lambda_j$$

using (9.4.33). Hence using (9.4.36), we see the eigenvalues of $\boldsymbol{B}$ satisfy (9.4.39) if

$$1 - 4\mu\lambda_k + 8\mu^2\lambda_k^2 + 4\mu^2\lambda_k \operatorname{tr}\boldsymbol{R} < 1$$

or

$$-\mu\lambda_k + 2\mu^2\lambda_k^2 + \mu^2\lambda_k \operatorname{tr}\boldsymbol{R} < 0$$

which implies that $\mu > 0$ and

$$2\mu < \frac{1}{\lambda_k + \operatorname{tr}\boldsymbol{R}} < \frac{1}{\operatorname{tr}\boldsymbol{R}}$$

because $\lambda_k > 0$ for all k. In conclusion, if the adaptation step μ satisfies the condition

$$0 < 2\mu < \frac{1}{\operatorname{tr}\boldsymbol{R}} \tag{9.4.40}$$

then the system (9.4.37) is stable and therefore the sequence $\theta(n)$ converges.

PROPERTY 9.4.2. When the stability condition (9.4.40) holds, the solution (9.4.37) of the difference equation (9.4.32) can be written as

$$\boldsymbol{\theta}(n) = \boldsymbol{B}^n[\boldsymbol{\theta}(0) - \boldsymbol{\theta}(\infty)] + \boldsymbol{\theta}(\infty) \tag{9.4.41}$$

where $\boldsymbol{\theta}(0)$ is the initial value and $\boldsymbol{\theta}(\infty)$ is the steady-state value of $\boldsymbol{\theta}(n)$.
Proof. Using the identity

$$\sum_{j=0}^{n-1}\boldsymbol{B}^j = (\boldsymbol{I} - \boldsymbol{B}^n)(\boldsymbol{I} - \boldsymbol{B})^{-1} = (\boldsymbol{I} - \boldsymbol{B})^{-1} - \boldsymbol{B}^n(\boldsymbol{I} - \boldsymbol{B})^{-1}$$

the solution (9.4.37) can be written as

$$\boldsymbol{\theta}(n) = \boldsymbol{B}^n[\boldsymbol{\theta}(0) - 4\mu^2 P_o(\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{\lambda}] + 4\boldsymbol{\mu}^2 P_o(\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{\lambda} \tag{9.4.42}$$

When the eigenvalues of $\boldsymbol{B}$ are inside the unit circle, we have

$$\lim_{n\to\infty}\theta(n) \triangleq \theta(\infty) = 4\mu^2 P_o(\boldsymbol{I} - \boldsymbol{B})^{-1}\lambda \tag{9.4.43}$$

because the first term converges to zero. Substituting (9.4.43) in (9.4.42), we obtain (9.4.41).

Evolution of the mean square error

We next express the MSE as a function of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$. Using (9.2.10) and (9.2.18), we have

$$e(n) = y(n) - \boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n) = e_o(n) - \tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n) \tag{9.4.44}$$

where $e_o(n)$ is the optimum filtering error and $\tilde{\mathbf{c}}(n)$ is the coefficient error vector. The (a priori) MSE of the adaptive filter at time n is

$$\begin{aligned} P(n) &\triangleq E\{|e(n)|^2\} \\ &= E\{|e_o(n)|^2\} - E\{\tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)e_o^*(n)\} - E\{e_o(n)\boldsymbol{x}^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}(n-1)\} \\ &\quad + E\{\tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}(n-1)\} \end{aligned} \tag{9.4.45}$$

Since $\tilde{\mathbf{c}}(n)$ is a random vector, the evaluation of the MSE (9.4.45) requires the correlation between $\boldsymbol{x}(n)$ and $\tilde{\mathbf{c}}(n-1)$. Using the independence assumptions A1 and A2, we see that the second and third terms in (9.4.45) become zero, as explained before, and the excess MSE is given by the last term

$$P_{\mathrm{ex}}(n) = E\{\tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}(n-1)\} \tag{9.4.46}$$

If we define the quantities

$$\boldsymbol{A} \triangleq \tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1) \qquad \text{and} \qquad \boldsymbol{B} \triangleq \boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}(n-1) \tag{9.4.47}$$

and notice that $\boldsymbol{AB} = \mathrm{tr}(\boldsymbol{AB})$ (because $\boldsymbol{AB}$ is a scalar) and $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$, we obtain

$$\begin{aligned} P_{\mathrm{ex}}(n) &= E\{\mathrm{tr}(\boldsymbol{AB})\} = E\{\mathrm{tr}(\boldsymbol{BA})\} = \mathrm{tr}(E\{\boldsymbol{BA}\}) \\ &= \mathrm{tr}(E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\}E\{\tilde{\boldsymbol{c}}(n-1)\tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\}) \end{aligned}$$

because expectation is a linear operation and $\boldsymbol{x}(n)$ and $\tilde{\boldsymbol{c}}(n-1)$ have been assumed statistically independent. Therefore, the excess MSE can be expressed as

$$P_{\mathrm{ex}}(n) = \mathrm{tr}[\boldsymbol{R\Phi}(n-1)] \tag{9.4.48}$$

where $\boldsymbol{\Phi}(n) = E\{\tilde{\boldsymbol{c}}(n)\tilde{\boldsymbol{c}}^{\mathrm{H}}(n)\}$ is the correlation matrix of the coefficient error vector. This expression simplifies to

$$P_{\mathrm{ex}}(n) = M\sigma_x^2\sigma_c^2 \tag{9.4.49}$$

if $\boldsymbol{R} = \sigma_x^2\boldsymbol{I}$ and $\boldsymbol{\Phi}(n) = \sigma_c^2\boldsymbol{I}$.

If $\boldsymbol{R}$ and $\boldsymbol{\Phi}(n)$ are both positive definite, relation (9.4.48) shows that $P_{\mathrm{ex}}(n) > 0$, that is, the MSE attained by the adaptive filter is larger than the optimum MSE P_o of the optimum filter (cost of adaptation).

Next we develop a difference equation for $P_{\mathrm{ex}}(n)$, using, for convenience, the principal coordinate system of the input correlation matrix $\boldsymbol{R}$. Since the trace of a matrix remains invariant under an orthogonal transformation, we have

$$P_{\mathrm{ex}}(n) = \mathrm{tr}[\boldsymbol{R\Phi}(n)] = \mathrm{tr}[\boldsymbol{\Lambda\Theta}(n)] = \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\theta}(n) \tag{9.4.50}$$

where the elements of $\boldsymbol{\lambda}$ are the eigenvalues of $\boldsymbol{R}$ and the elements of $\boldsymbol{\theta}(n)$ are the diagonal elements of $\boldsymbol{\Theta}(n)$.

Since the most often observable and important quantity for the operation of an adaptive filter is the MSE, we use our previous results to determine the value of MSE as a function of n, that is, the learning curve of the LMS adaptive filter. To this end, we use the orthogonal decomposition $\boldsymbol{B} = \boldsymbol{Q}(\boldsymbol{B})\boldsymbol{\Lambda}(\boldsymbol{B})\boldsymbol{Q}^H(\boldsymbol{B})$ to express $\boldsymbol{B}^n$ as

$$\boldsymbol{B}^n = \boldsymbol{Q}(\boldsymbol{B})\boldsymbol{\Lambda}^n(\boldsymbol{B})\boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{B}) = \sum_{k=1}^{M} \lambda_k^n(\boldsymbol{B})\boldsymbol{q}_k(\boldsymbol{B})\boldsymbol{q}_k^{\mathrm{H}}(\boldsymbol{B}) \tag{9.4.51}$$

where $\lambda_k(\boldsymbol{B})$ are the eigenvalues and $\boldsymbol{q}_k(\boldsymbol{B})$ are the eigenvectors of matrix $\boldsymbol{B}$. Substituting (9.4.41) and (9.4.51) into (9.4.50) and recalling that $P(n) = P_o + P_{\mathrm{ex}}(n)$, we obtain

$$P(n) = P_o + P_{\mathrm{tr}}(n) + P_{\mathrm{ex}}(\infty) \tag{9.4.52}$$

where $P_{ex}(\infty)$ is termed the *steady-state excess MSE* and

$$P_{tr}(n) \triangleq \sum_{k=1}^{M} \gamma_k(\boldsymbol{R},\boldsymbol{B})\lambda_k^n(\boldsymbol{B}) \tag{9.4.53}$$

is termed the *transient MSE* because it dies out exponentially when $0<\lambda_k(\mathbf{B})<1,\ 1\le k\le M$. The constants

$$\gamma_k(\boldsymbol{R},\boldsymbol{B}) \triangleq \boldsymbol{\lambda}^{T}(\boldsymbol{R})\boldsymbol{q}_k(\boldsymbol{B})\boldsymbol{q}_k^{H}(\boldsymbol{B})[\boldsymbol{\theta}(0)-\boldsymbol{\theta}(\infty)] \tag{9.4.54}$$

are determined by the eigenvalues $\lambda_k(\boldsymbol{R})$ of matrix $\boldsymbol{R}$ and the eigenvectors $\boldsymbol{q}_k(\boldsymbol{B})$ of matrix $\boldsymbol{B}$. Since the minimum MSE P_o is available, we need to determine the steady-state excess MSE $P_{ex}(\infty)$.

PROPERTY 9.4.3. When the LMS adaptive algorithm converges, the steady-state excess MSE is given by

$$P_{ex}(\infty) = P_o \frac{C(\mu)}{1-C(\mu)} \tag{9.4.55}$$

where

$$C(\mu) \triangleq \sum_{k=1}^{M} \frac{\mu\lambda_k}{1-2\mu\lambda_k} \tag{9.4.56}$$

and λ_k are the eigenvalues of the input correlation matrix.

Proof. Using (9.4.32) and (9.4.35), we obtain the difference equation

$$\theta_k(n) = \rho_k\theta_k(n-1) + 4\mu^2\lambda_k P_{ex}(n-1) + 4\mu^2 P_o\lambda_k \tag{9.4.57}$$

When (9.4.40) holds, (9.4.57) attains the following steady-state form

$$\theta_k(\infty) = \rho_k\theta_k(\infty) + 4\mu^2\lambda_k P_{ex}(\infty) + 4\mu^2 P_o\lambda_k$$

whose solution, in conjunction with (9.4.36), gives

$$\theta_k(\infty) = \mu\frac{P_o + P_{ex}(\infty)}{1-2\mu\lambda_k}$$

and

$$P_{ex}(\infty) = \sum_{k=1}^{M}\lambda_k\theta_k(\infty) = [P_o + P_{ex}(\infty)]\sum_{k=1}^{M}\frac{\mu\lambda_k}{1-2\mu\lambda_k}$$

Solving the last equation for $P_{ex}(\infty)$, we obtain (9.4.55) and (9.4.56).

Solving (9.4.55) for $C(\mu)$ gives

$$C(\mu) = \frac{P_{ex}(\infty)}{P_o + P_{ex}(\infty)} \tag{9.4.58}$$

which implies that

$$0 < C(\mu) < 1 \tag{9.4.59}$$

because P_o and $P_{ex}(\infty)$ are positive quantities. It has been shown that (9.4.59) leads to the tighter bound $0<2\mu<2/(3\,\mathrm{tr}\,\boldsymbol{R})$ for the adaptation step μ (Horowitz and Senne 1981; Feuer and Weinstein 1985). Therefore, convergence in the MSE imposes a stronger constraint on the step size μ than does (9.4.40), which ensures convergence in the mean.

9.4.3 Summary and Design Guidelines

There are many theoretical and simulation analyses of the LMS adaptive algorithm under a variety of assumptions. In this book, we have focused on results that help us to understand its operation and performance and to develop design guidelines for its practical application. The operation and performance of the LMS adaptive filter are determined by its stability and the properties of its learning curve, which shows the evolution of the MSE as a function of time. The MSE produced by the LMS adaptive algorithm consists of three components [see (9.4.52)]

$$P(n) = P_o + P_{tr}(n) + P_{ex}(\infty)$$

where P_o is the optimum MSE, $P_{tr}(n)$ is the transient MSE, and $P_{ex}(\infty)$ is the steady-state excess MSE. This equation provides the basis for understanding and evaluating the operation of the LMS adaptive algorithm in a stationary SOE. For convenience, the LMS adaptive filtering algorithm is summarized in Table 9.3.

TABLE 9.3
Summary of the LMS algorithm.

Design parameters
$\mathbf{x}(n)$ = input data vector at time n
$y(n)$ = desired response at time n
$\mathbf{c}(n)$ = filter coefficient vector at time n
M = number of coefficients
μ = step-size parameter
$0<\mu \ll \frac{1}{\sum_{k=1}^{M} E\{\lvert x_k(n)\rvert^2\}}$
Initialization
$\mathbf{c}(-1)=\mathbf{x}(-1)=0$
Computation
For $n=0,1,2,\cdots,$ compute
$\hat{y}(n)=\mathbf{c}^H(n-1)\mathbf{x}(n)$
$e(n)=y(n)-\hat{y}(n)$
$\mathbf{c}(n)=\mathbf{c}(n-1)+2\mu\mathbf{x}(n)e^*(n)$

Stability. The LMS adaptive filter converges in the mean-square sense, that is, the transient MSE dies out, if the adaptation step μ satisfies the condition

$$0<2\mu<\frac{K}{\operatorname{tr}\boldsymbol{R}} \tag{9.4.60}$$

where $\operatorname{tr}\boldsymbol{R}$ is the trace of the input correlation matrix and K is a constant that depends weakly on the statistics of the input data vector. For example, when $\boldsymbol{x}(n)\sim\mathcal{N}(0,\boldsymbol{R})$, we proved that K=1 or 2/3. In addition, this condition ensures that on average the LMS adaptive filter converges to the optimum filter. We stress that in most practical applications, where the independence assumption does not hold, the *step size* μ *should be much smaller than* $K/\operatorname{tr}\boldsymbol{R}$. Therefore, the exact value of K is not important in practice.

Rate of convergence. The transient MSE dies out exponentially without exhibiting any oscillations. This follows from (9.4.53) because when μ satisfies (9.4.40), the eigenvalues of matrix $\boldsymbol{B}$ are positive and less than 1. The *settling time*, that is, the time taken for the transients to die out, is proportional to the average time constant

$$\tau_{\text{lms,av}}=\frac{1}{\mu\lambda_{\text{av}}} \tag{9.4.61}$$

where $\lambda_{\text{av}}=(\sum_{k=1}^{M}\lambda_k)/M$ is the average eigenvalue of $\boldsymbol{R}$ (Widrow et al. 1976). The quantity $P_{\text{tr}}^{\text{total}}=\sum_{n=0}^{\infty}P_{\text{tr}}(n)$, which provides the total transient MSE, can be used as a measure for the speed of adaptation. When $\mu\lambda_k \ll 1$ (see Problem 9.12), we have

$$P_{\text{tr}}^{\text{total}} \triangleq \sum_{n=0}^{\infty}P_{\text{tr}}(n)\simeq\frac{1}{4\mu}\sum_{k=1}^{M}\Delta\theta_k(0) \tag{9.4.62}$$

where $\Delta\theta_k(0)$ is the initial distance of a coefficient from its optimum setting measured in principal coordinates. As is intuitively expected, *the smaller the step size and the farther the initial coefficients are from their optimum settings, the more iterations it takes for the LMS algorithm to converge*. Furthermore, from the discussion in Section 9.3, it follows that the LMS algorithm will converge faster if the contours of the error surface are circles, that is, when the input correlation matrix is $\boldsymbol{R}=\sigma_x^2\boldsymbol{I}$.

Steady-state excess MSE. The excess MSE after the adaptation has been completed (i.e., the steady-state value) is given by (9.4.55). When $\mu\lambda_k \ll 1$, we may approximate (9.4.55) as follows

$$P_{\text{ex}}(\infty) \simeq P_o \frac{\mu \operatorname{tr} \boldsymbol{R}}{1-\mu \operatorname{tr} \boldsymbol{R}}$$

which allows a much easier interpretation. Solving for $\mu \operatorname{tr} \boldsymbol{R}$, we obtain $\mu \operatorname{tr} \boldsymbol{R} \simeq P_{\text{ex}}(\infty)/[P_{\text{ex}}(\infty)+P_o]$ which implies that $0 < \mu \operatorname{tr} \boldsymbol{R} < 1$. Since $\mu \operatorname{tr} \boldsymbol{R} \ll 1$, we often use the approximation

$$P_{\text{ex}}(\infty) \simeq \mu P_o \operatorname{tr} \boldsymbol{R} \tag{9.4.63}$$

which implies that $P_{\text{ex}}(\infty) \ll P_o$, that is, for small values of the step size the excess MSE is much smaller than the optimum MSE. Note that the presence of the irreducible error $e_o(n)$ prevents perfect adaptation as $n \to \infty$ because $P_o > 0$.

Speed versus quality of adaptation. From the previous discussion we see that there is a tradeoff between rate of convergence (speed of adaptation) and steady-state excess MSE (quality of adaptation, or accuracy of the adaptive filter). The first requirement for an adaptive filter is stability, which is ensured by choosing μ to satisfy (9.4.60). Within this range, decreasing μ to reduce the desired level of misadjustment, according to (9.4.63), decreases the speed of convergence; see (9.4.62). Conversely, if μ is increased to increase the speed of convergence; this results in an increase in misadjustment. This tradeoff between speed of convergence and misadjustment is a fundamental feature of the LMS algorithm.

FIR filters. In this case, the input is a stationary process $x(n)$ with a Toeplitz correlation matrix $\boldsymbol{R}$. Therefore, we have

$$\operatorname{tr} \boldsymbol{R} = Mr(0) = ME\{|x(n)|^2\} = MP_x \tag{9.4.64}$$

where MP_x is called the *tap input power*. Substituting (9.4.40) into (9.4.64), we obtain

$$0 < 2\mu < \frac{1}{MP_x} = \frac{1}{\text{tap input power}} \tag{9.4.65}$$

which shows that the selection of the step size depends on the input power. Using (9.4.63) and (9.4.64), we see that misadjustment M is given by

$$M = \frac{P_{\text{ex}}(\infty)}{P_o} \simeq \mu MP_x \tag{9.4.66}$$

which shows that for given M and P_x the value of misadjustment is proportional to μ. We emphasize that the misadjustment provides a measure of how close an LMS adaptive filter is to the corresponding optimum filter.

The statistical properties of the SOE, that is, the correlation of the input signal and the cross-correlation between input and desired response signals, play a key role in the performance of the LMS adaptive filter.

- First, we should make sure that the relation between $x(n)$ and $y(n)$ can be accurately modeled[①] or lack of correlation between $x(n)$ and $y(n)$ increases the magnitude of the irreducible error. If M is very large, we may want to use a pole-zero IIR filter (Shynk 1989; Treichler et al. 1987). If the relationship between $x(n)$ and $y(n)$ is nonlinear, we certainly need a nonlinear filtering structure (Mathews 1991).
- The LMS algorithm uses a "noisy" instantaneous estimate of the gradient vector. However, when the correlation between input and desired response is weak, the algorithm should make more cautious steps ("wait and average"). Such algorithms update their coefficients every L samples, using all samples between successive updatings to determine the gradient (gradient averaging).
- The eigenvalue structure of $\boldsymbol{R}$ as measured by its eigenvalue spread $(\lambda_{\max}/\lambda_{\min})$ or equivalently by the *spectral flatness measure* (*SFM*) (see Section 3.1) has a strong effect on the rate of convergence of the LMS algorithm. In general, the rate of convergence decreases as the eigenvalue spread increases, that is, as the contours of the cost function become more elliptical, or equivalently the input spectrum becomes more nonwhite.

Normalized LMS algorithm. According to (9.4.60), the selection of μ in practical applications is complicated because the power of the input signal either is unknown or varies with time. This problem can be addressed by using

---

[①] by a linear FIR filter with M coefficients. Inadequacy of the FIR structure, output observation noise.

the *normalized LMS (NLMS)* algorithm [see (9.4.11)]

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \frac{\tilde{\mu}}{E_M(n)} \mathbf{x}(n) e^*(n) \tag{9.4.67}$$

where $E_M(n) = \|\mathbf{x}(n)\|^2$ and $0 < \tilde{\mu} < 1$. It can be shown that the NLMS algorithm converges in the mean square if $0 < \tilde{\mu} < 1$ (Rupp 1993; Slock 1993), which makes the selection of the step size $\tilde{\mu}$ much easier than the selection of μ in the LMS algorithm.

For FIR filters, the quantity $E_M(n)$ provides an estimate of $ME\{|x(n)|^2\}$ and can be computed recursively by using the sliding-window formula

$$E_M(n) = E_M(n-1) + |x(n)|^2 - |x(n-M)|^2 \tag{9.4.68}$$

where $E_M(-1) = 0$ or a first-order recursive filter estimator. In practice, to avoid division by zero, if $x(n) = \mathbf{0}$, we set $E_M(n) = \delta + \|\mathbf{x}(n)\|^2$, where δ is a small positive constant.

Other approaches and analyses. The analysis of the LMS algorithm presented in this section is simple, clarifies its performance, and provides useful design guidelines. However, there are many other approaches, which are beyond the scope of this book, that differ in terms of complexity, accuracy, and objectives. Major efforts to remove the independence assumption and replace it with the more realistic statistically dependent input assumption are documented in Macchi (1995), Solo (1997), and Butterweck (1995) and the references therein. Convergence analysis of the LMS algorithm using the *stochastic approximation approach* and a deterministic approach using the *method of ordinary differential equations* are discussed in Solo and Kong (1995), Sethares (1993), and Benveniste et al. (1987). Other types of analyses deal with the determination of the probability densities and the probability of large excursions of the adaptive filter coefficients for various types of input signals (Rupp 1995). The analysis of the convergence properties of the LMS algorithm and its variations is still an active area of research, and new results appear continuously.

9.4.4 Applications of the LMS Algorithm

We now discuss three practical applications in which the LMS algorithm has made a significant impact. In the first case, we consider the previously discussed linear prediction problem and compare the performance of the LMS algorithm with that of the SDA. Table 9.4 provides a summary of the key differences between the SDA and the LMS algorithms. In the second case, we study echo cancelation in full-duplex data transmission, which employs the LMS algorithm in its implementation. In the third case, we discuss the application of adaptive equalization, which is used to minimize intersymbol interference (ISI) in a dispersive channel environment.

TABLE 9.4
Comparison between the SDA and LMS algorithms.

SDA	LMS
Deterministic algorithm:	Stochastic algorithm:
$\lim_{n\to\infty} \mathbf{c}(n) = \mathbf{c}_o$	$\lim_{n\to\infty} E\{\mathbf{c}(n)\} = \mathbf{c}_o$
If converges, it terminates to $\mathbf{c}_o$	If converges, it fluctuates about $\mathbf{c}_o$
	The size of fluctuations is proportional to μ
Noiseless gradient estimate	Noisy gradient estimate
Deterministic steps	Random steps
We can only compare the ensemble average behavior of LMS with the SDA.	

Linear prediction

In Example 9.3.1, the AR(2) model given in (9.3.25) was considered, and the SDA was used to determine the corresponding linear predictor coefficients. We also analyzed the performance of the SDA. In the following example, we perform a similar acquisition of predictor coefficients using the LMS algorithm, and we study the effects of the eigenvalue spread of the input correlation matrix on the convergence of the LMS adaptive algorithm when it is used to update the coefficients.

EXAMPLE 9.4.1. The second-order system in (9.3.25) is repeated here, which generates the signal $x(n)$:

$$x(n) + a_1 x(n-1) + a_2 x(n-2) = \omega(n)$$

where $\omega(n) \sim \text{WGN}(0, \sigma_\omega^2)$ and where the coefficients are selected from Table 9.2 for two different eigenvalue spreads. A Gaussian pseudorandom number generator was used to obtain 1000 realizations of $x(n)$ using each set of parameter values given in Table 9.2. These sample realizations were used for statistical analysis.

The second-order LMS adaptive predictor with coefficients $\mathbf{c}(n) = [c_1(n)\ c_2(n)]^T$ is given by [see (9.4.12)]

$$e(n) = x(n) - c_1(n-1)x(n-1) - c_2(n-2)x(n-2) \qquad n \geq 0$$

$$c_1(n) = c_1(n-1) + 2\mu e(n)x(n-1)$$

$$c_2(n) = c_2(n-1) + 2\mu e(n)x(n-2)$$

where μ is the step-size parameter. The adaptive predictor was initialized by setting $x(-1) = x(-2) = 0$ and $c_1(-1) = c_2(-1) = 0$. The above adaptive predictor was implemented with $\mu = 0.04$, and the predictor coefficients as well as the MSE were recorded for each realization. These quantities were averaged to study the behavior of the LMS algorithm. These calculations were repeated for $\mu = 0.01$.

In Figure 9.20 we show several plots obtained for $\chi(\boldsymbol{R}) = 1.22$. In plot (*a*) we show the ensemble averaged trajectory $\{\mathbf{c}(n)\}_{n=0}^{150}$ superimposed on the MSE contours. A trajectory of a simple realization is also shown to illustrate its randomness. In plot (*b*) the $\mathbf{c}(n)$ learning curve for the averaged value as well as for one single realization is shown. In plot (*c*) the corresponding learning curves for the MSE are depicted. Finally, in plot (*d*) we show the effect of step size μ on the MSE learning curve. Similar plots are shown in Figure 9.21 for $\chi(\boldsymbol{R}) = 10$.

Several observations can be made from these plots:

- The trajectories and the learning curves for a simple realization are clearly random or "noisy," while the averaging over the ensemble clearly has a smoothing effect.
- The averaged quantities (coefficients and the MSE) converge to the true values, and this convergence rate is in accordance with theory.
- The rate of convergence of the LMS algorithm depends on the step size μ. The smaller the step size, the slower the rate.
- The rate of convergence also depends on the eigenvalue spread $\chi(\boldsymbol{R})$. The larger the spread, the slower the rate. For $\chi(\mathbf{R}) = 1.22$ the algorithm converges in about 150 steps while for $\chi(\boldsymbol{R}) = 10$ it requires about 500 steps.

Clearly these observations compare well with the theory.

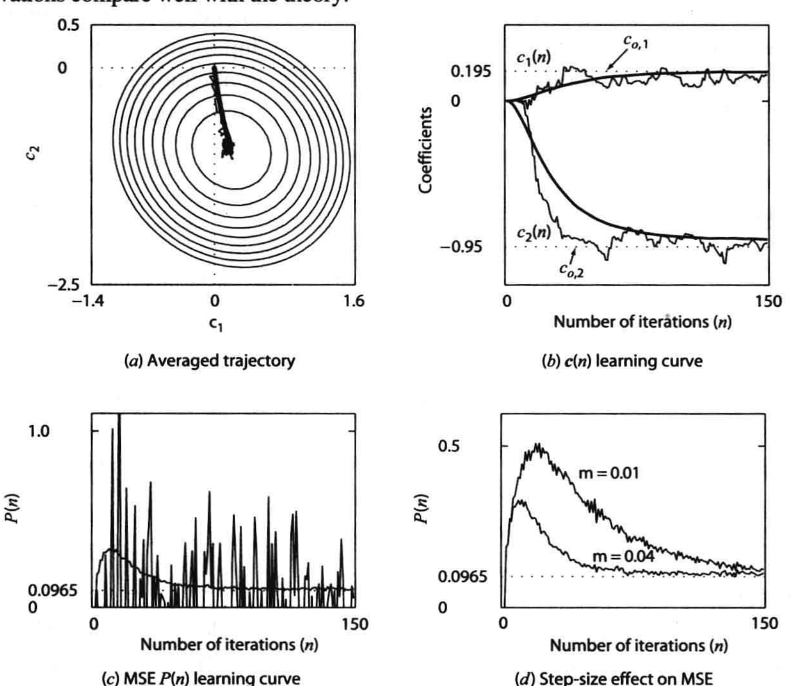


FIGURE 9.20

Performance curves for the LMS used in the linear prediction problem with step-size parameter $\mu = 0.04$ and eigenvalue spread $\chi(\mathbf{R}) = 1.22$.

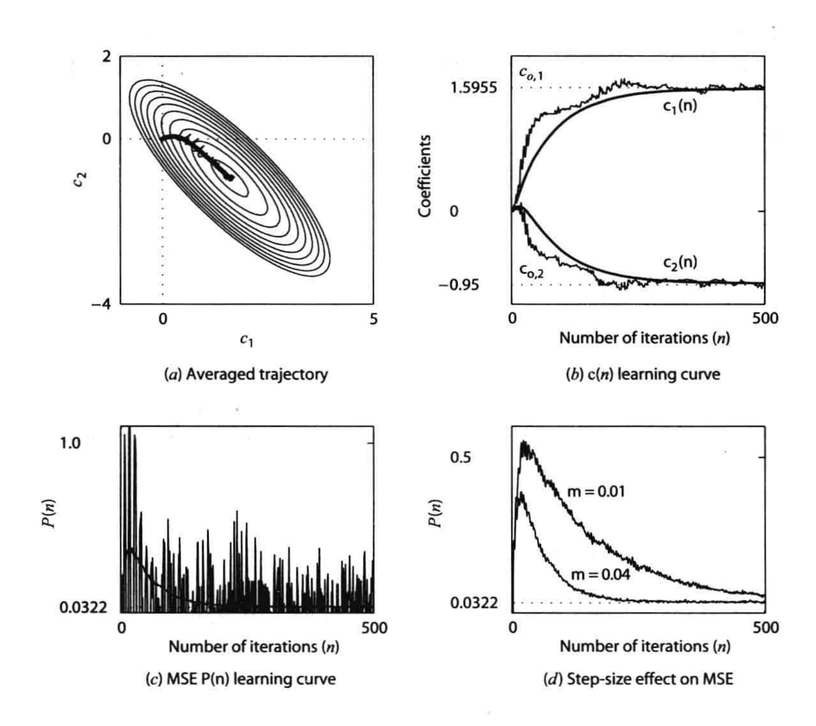


FIGURE 9.21
Performance curves for the LMS used in the linear prediction problem with step-size parameter $\mu = 0.04$ and eigenvalue spread $\chi(\mathbf{R}) = 10$.

Echo cancelation in full-duplex data transmission

Figure 9.22 illustrates a system that achieves simultaneous data transmission in both directions (full-duplex) over two-wire circuits using the special two-wire to four-wire interfaces (called *hybrid couplers*) that exist in any telephone set. Although the hybrid couplers are designed to provide perfect isolation between transmitters and receivers, this is not the case in practical systems. As a result, (1) one part of the transmitted signal leaks through the near-end hybrid to its own receiver (near-end echo), and (2) another part is reflected by the far-end hybrid and ends up at its own receiver (far-end echo). The combined echo signal, which can be 30 dB stronger than the signal received from the other end, increases the number of errors. We note that in contrast with acoustic echo cancelation, the delay of echoes in data transmission is immaterial.

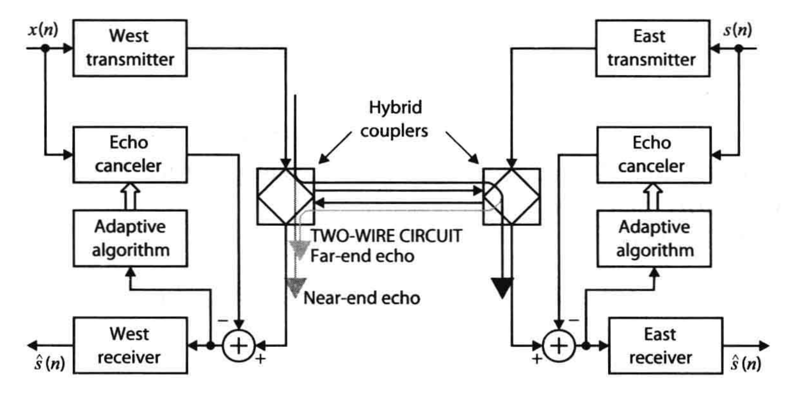


FIGURE 9.22
Model of a full-duplex data transmission system that uses an echo canceler in the modems.

The best way to address this problem is to form a replica of the echo and then subtract it from the incoming signal. We can model the echoes as the result of an "echo" path between the transmitter and the receiver. For baseband data transmission this echo path is basically linear and varies very slowly with time. Therefore, we can obtain a replica of the echo signal using an FIR LMS adaptive filter (*echo canceler*), as shown in Figure 9.22. The

inclusion of the transmitter in the echo path, as long as it involves linear operations, simplifies the implementation and improves the speed of adaptation because the input is an IID binary data sequence of values $+1$ and -1 with equal probability (Verhoeckx et al. 1979).

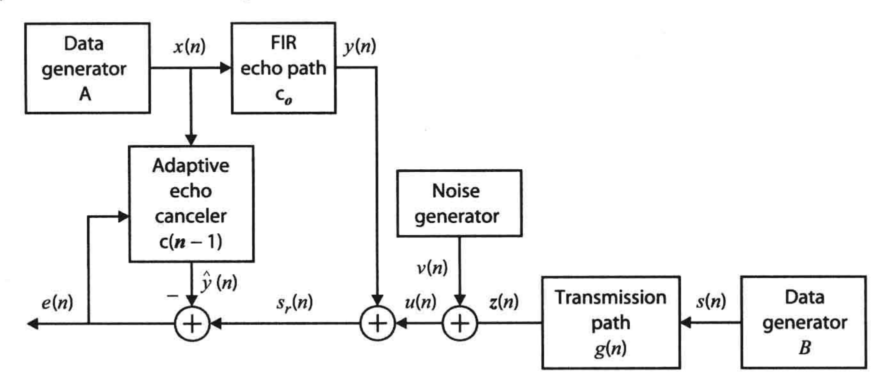


FIGURE 9.23
Block diagram of a system for investigating the performance of adaptive echo canceler.

Referring to Figure 9.23, if we assume that the echo path has an FIR impulse response, the echo signal is given by

$$y(n) = \boldsymbol{c}_o^{\mathrm{T}} \boldsymbol{x}(n) \tag{9.4.69}$$

where
$$c_o = [c_o(0)\ c_o(1)\ \cdots\ c_o(M-1)]^{\mathrm{T}}$$

If $g(n)$ is the impulse response of the transmission path from the far-end transmitter to the near-end receiver, the received signal is given by

$$s_r(n) = y(n) + z(n) + v(n) \triangleq y(n) + u(n) \tag{9.4.70}$$

$$z(n) = \sum_{k=0}^{\infty} g(k)s(n-k)$$

where $s(n)$ is the transmitted data signal and $v(n) \sim \mathrm{WGN}(0, \sigma_v^2)$ is additive noise. The signal $u(n) = z(n) + v(n)$ represents the "uncancelable" signal because it cannot be removed by the canceler.

The LMS adaptive echo canceler is given by

$$\hat{y}(n) = \boldsymbol{c}^{\mathrm{T}}(n-1)\boldsymbol{x}(n) \tag{9.4.71}$$

$$e(n) = y(n) - \hat{y}(n) \tag{9.4.72}$$

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + 2\mu e(n)\boldsymbol{x}(n) \tag{9.4.73}$$

where μ is the adaptation step size. The adaptive filter takes advantage of the fact that $x(n)$ is correlated with $y(n)$ but uncorrelated with $s(n)$ and $v(n)$.

The residual (uncanceled) echo is

$$e_r(n) \triangleq y(n) - \hat{y}(n) = [\boldsymbol{c}_o - \boldsymbol{c}(n-1)]^{\mathrm{T}} \boldsymbol{x}(n) \triangleq -\tilde{\boldsymbol{c}}^{\mathrm{T}}(n-1)\boldsymbol{x}(n) \tag{9.4.74}$$

and if we assume that $\tilde{\boldsymbol{c}}(n-1)$ and $\boldsymbol{x}(n)$ are independent, then

$$P_r(n) = E\{e_r^2(n)\} = E\{\tilde{\mathbf{c}}^{\mathrm{T}}(n-1)\tilde{\mathbf{c}}(n-1)\}$$

because $\boldsymbol{R} = E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)\} = \boldsymbol{I}$. Using (9.4.69), (9.4.71), and (9.4.72), we can easily show that

$$\tilde{\boldsymbol{c}}(n) = \tilde{\boldsymbol{c}}(n-1) - 2\mu\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)\tilde{\boldsymbol{c}}(n-1) + 2\mu\boldsymbol{x}(n)u(n) \tag{9.4.75}$$

If we premultiply (9.4.75) by its transpose and take the mathematical expectation, we obtain

$$P_r(n+1) = (1-4\mu+4\mu^2 M)P_r(n) + 4\mu M\sigma_u^2 \tag{9.4.76}$$

using the independence assumption and the relation $\mathbf{x}^T(n)\mathbf{x}(n) = M$. The solution of (9.4.76), in terms of the residual echo ratio $P_r(n)/\sigma_u^2$, is

$$\frac{P_r(n)}{\sigma_u^2} = (1-4\mu+4\mu^2 M)^n \left[\frac{P_r(0)}{\sigma_u^2} - \frac{\mu M}{1-\mu M}\right] + \frac{\mu M}{1-\mu M} \tag{9.4.77}$$

and describes completely the operation of the LMS adaptive echo canceler. Indeed, we draw the following conclusions:

1. The algorithm converges if

$$|1-4\mu+4\mu^2 M| < 1 \qquad \text{or} \qquad 0 < \mu < \frac{1}{M} \tag{9.4.78}$$

which agrees with (9.4.40) because $\operatorname{tr} \mathbf{R} = M$.

2. After convergence we have

$$P_r(\infty) = \frac{\mu M}{1-\mu M}\sigma_u^2 \simeq \mu M \sigma_u^2 \tag{9.4.79}$$

which again is in agreement with (9.4.63).

3. If $P_r(n)/\sigma_u^2 \gg \mu M/(1-\mu M)$, we have

$$\frac{P_r(n)}{P_r(0)} \simeq (1-4\mu+4\mu^2 M)^n \tag{9.4.80}$$

which can be used to find out how many iterations are required for a given echo reduction. For example, we can easily show that to achieve a 20-dB echo reduction requires $n_{20} \simeq 1.15/\mu$ iterations.

From the previous discussion, it should be clear that the step size μ plays a crucial role in the performance of the adaptive echo canceler because it determines both the rate of convergence and the minimum residual echo cancelation that can be attained. Furthermore, we clearly see the tradeoff between fast adaptation and residual echo power.

EXAMPLE 9.4.2. Consider the system shown in Figure 9.23 for investigating the performance of the LMS algorithm in adaptive echo cancelation and to verify the above conclusions. The data generators *A* (in modem *A*) and *B* (in modem *B*) output symbols +1 or −1 with equal probability (i.e., Bernoulli sequence). The FIR filter following data generator *A* models the echo path, which is assumed to be

$$c_o(n) = -\frac{5}{3}\left(\frac{1}{2}\right)^n + \frac{8}{3}\left(\frac{4}{5}\right)^n \qquad 0 \le n \le M-1$$

where $M = 20$ is the total length of the echo path. The filter following data generator *B* models the transmission path between the far-end transmitter and the near-end receiver, which we will assume to be

$$g(n) = \frac{4}{5}\left(\frac{3}{5}\right)^n \quad n \ge 0$$

The noise generator is a white Gaussian source with $\sigma_v^2 = 1$ and models the transmission noise. Using the equations $\sigma_y^2 = \sum_{k=0}^{N-1} c_o^2(k)$ and $\sigma_u^2 = \sum_{k=0}^{\infty} g^2(k) + \sigma_v^2$, we scale $u(n)$ so that $10\log(\sigma_y^2/\sigma_u^2) = 30$ dB. The adaptive echo canceler employs the LMS algorithm with $\mathbf{c}(0) = \mathbf{0}$. We perform Monte Carlo simulations on this system. Figure 9.24 shows the residual echo ratio $P_r(n)/\sigma_u^2$ evaluated by ensemble averaging over 200 independent trials of the experiment, for two different step sizes in the LMS algorithm [which satisfy (9.4.78)], superimposed on the corresponding theoretical curves computed by using (9.4.79) and (9.4.80). Clearly, the simulations support the theoretical results quite accurately. More detailed discussions of adaptive echo cancelation techniques for both baseband and passband data transmission systems can be found in Gitlin et al. (1992) and in Ling (1993a).

Adaptive equalization

When data are transmitted below 2400 bits/s, the **ISI** is relatively small and does not pose a problem in the

operation of a modem. However, for high-speed communication over 2400 bits/s, an equalizer is needed in the modem to compensate for the channel distortion. Since channel characteristics are generally unknown and time-varying, an adaptive algorithm is required that leads to adaptive equalization. Figure 9.25 describes an application of adaptive filtering to adaptive channel equalization. Initially, coefficients of the equalizer are adjusted, by means of the LMS algorithm, by transmitting a known training sequence of short duration. After this short training period, the actual data sequence $\{y(n)\}$ is transmitted. The slow variation in channel characteristics is then continuously tracked by adjusting coefficients of the equalizer, using the decisions in place of the known training sequence. This approach works well when decision errors are infrequent.

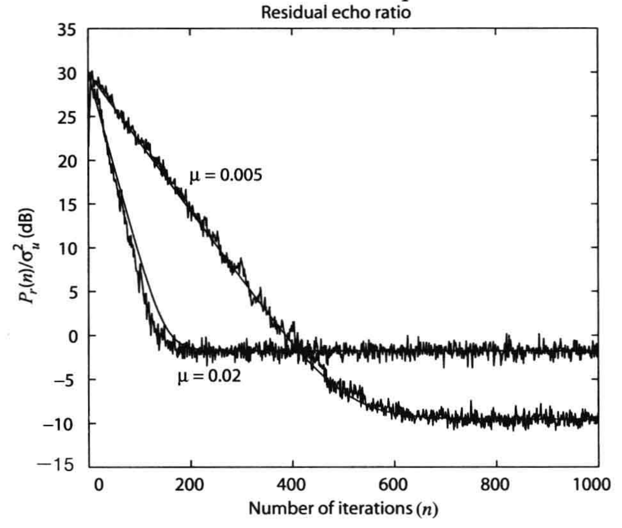


FIGURE 9.24

Performance analysis of the LMS algorithm in the adaptive echo cancelation that clearly shows the tradeoff between rate of convergence and residual echo power.

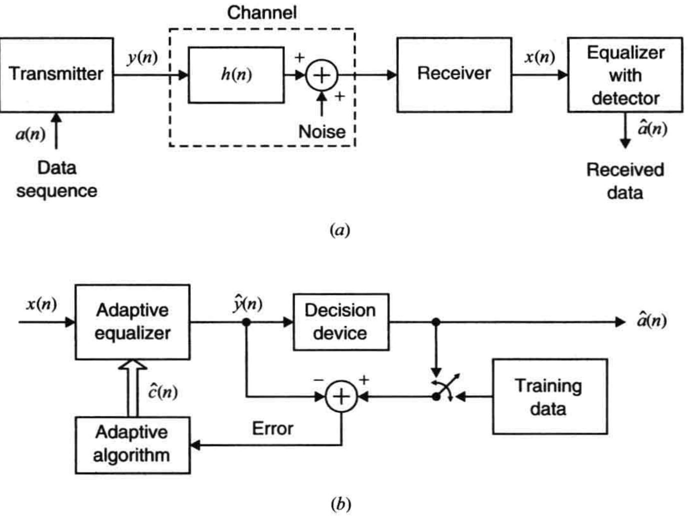


FIGURE 9.25

Model of an adaptive equalizer in a data transmission system.

EXAMPLE 9.4.3. Figure 9.26 shows the block diagram of the system used in the experimental investigation of the performance of the LMS algorithm used in the adaptive equalizer. The data source generates Bernoulli sequence $\{y(n)\}$ with symbols +1 and −1 having zero mean and unit variance. The channel following the source is modeled by the raised cosine impulse response

$$h(n) = \begin{cases} 0.5\left\{1+\cos\left[\dfrac{2\pi}{W}(n-2)\right]\right\} & n = 1,\ 2,\ 3 \\ 0 & \text{otherwise} \end{cases} \tag{9.4.81}$$

where parameter W is used to control the amount of channel distortion. The amount of channel distortion increases with W. The random noise generator outputs white Gaussian sequence $v(n)$ which models the noise in the channel. The equalizer input is

$$x(n) = \sum_{k=1}^{3} h(k)y(n-k) + v(n) \tag{9.4.82}$$

Since $y(n)$ is an independent sequence and since $v(n)$ is uncorrelated with $y(n)$, the maximum lag that produces nonzero correlation is 2. Thus the correlation of $x(n)$ is given by

$$r_x(0) = h^2(1) + h^2(2) + h^2(3) + \sigma_v^2$$
$$r_x(1) = h(1)h(2) + h(2)h(3)$$
$$r_x(2) = h(1)h(3)$$

from which an $M \times M$ autocorrelation matrix $\boldsymbol{R}$ can be constructed for an equalizer of length M. Clearly, parameter W also controls the eigenvalues of $\boldsymbol{R}$ and hence the ratio $\chi(\boldsymbol{R})$. Here we study the performance of the corresponding LMS adaptive equalizer.

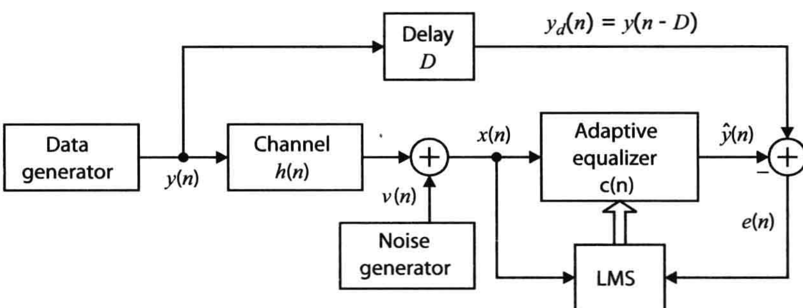


FIGURE 9.26
Block diagram of a system for investigating the performance of an adaptive equalizer.

The training signal $y(n)$ is delayed by an amount equal to the combined delay introduced by the channel and the equalizer for the desired signal. The impulse response $h(n)$ in (9.4.81) is symmetric with respect to $n = 2$, and assuming that the equalizer is a linear-phase FIR filter, the total delay is equal to $\Delta = (M-1)/2 + 2$. The error signal $e(n) = y(n-\Delta) - \hat{y}(n)$ is used along with $x(n)$ to implement the LMS algorithm in the adaptive equalizer with $\mathbf{c}(0) = 0$. We performed Monte Carlo simulations using 100 realizations of random sequences with $M = 11; \Delta = 7; \sigma_v^2 = 0.001$; $W = 2.9$ and $W = 3.5$; and $\mu = 0.01$, 0.04, and 0.08 The results are shown in Figures 9.27 and 9.28.

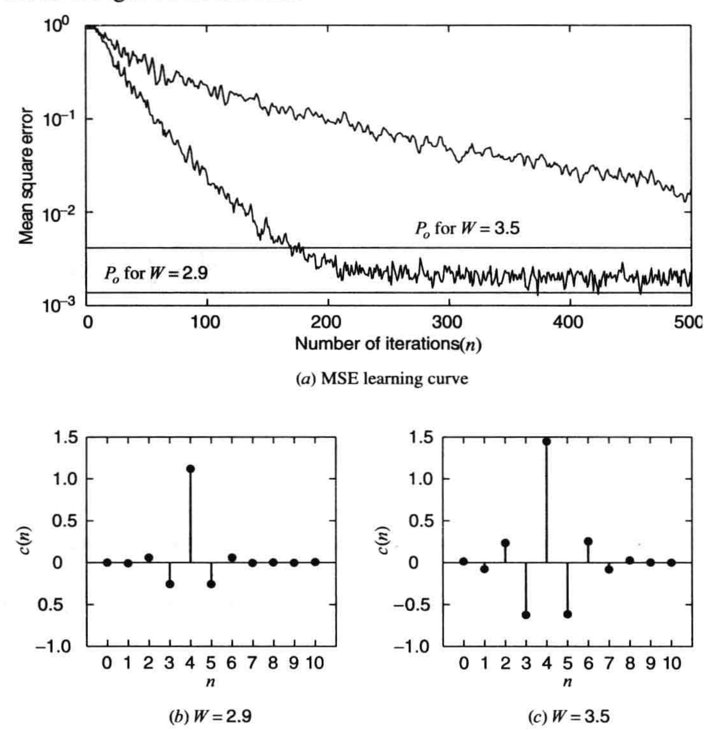


FIGURE 9.27
Performance analysis curves of the LMS algorithm in the adaptive equalizer: $\mu = 0.04$.

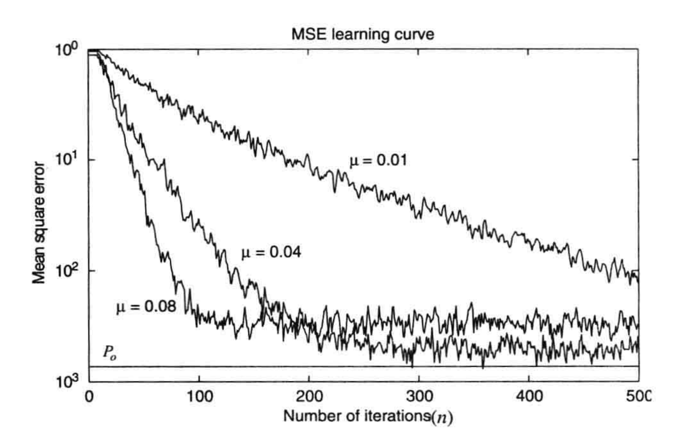


FIGURE 9.28
MSE learning curves of the LMS algorithm in the adaptive equalizer: $W = 2.9$.

Effect of eigenvalue spread. Performance plots of the LMS algorithm for $W = 2.9$ and $W = 3.5$ are shown in Figure 9.27. In plot (*a*) we depict MSE learning curves from which we observe that the convergence rate of the MSE decreases with *W* [or equivalently with increase in $\chi(\boldsymbol{R})$], which is to be expected. The steady-state error, on the other hand, increases with *W*. In plots (*b*) and (*c*) we show the ensemble averaged equalizer coefficients. Clearly, the responses are symmetric with respect to $n = 5$ as assumed. Also equalizer coefficients converge to different inverses due to changes in the channel characteristics.

Effect of step size μ. In Figure 9.28 we show the MSE learning curves obtained for $W = 2.9$ and with three different step-size parameter values of 0.01, 0.04, and 0.08. It indicates that μ affects the rate of convergence as well as the steady-state value. For $\mu = 0.08$, the algorithm converges in about 100 iterations but has higher steady-state value than the case for $\mu = 0.04$, which requires about 275 iterations for convergence. For $\mu = 0.01$ more than 500 iterations are needed. Finally, Figure 9.29 shows sample realizations of the transmitted, received, and equalized sequences using the discussed LMS equalizer.

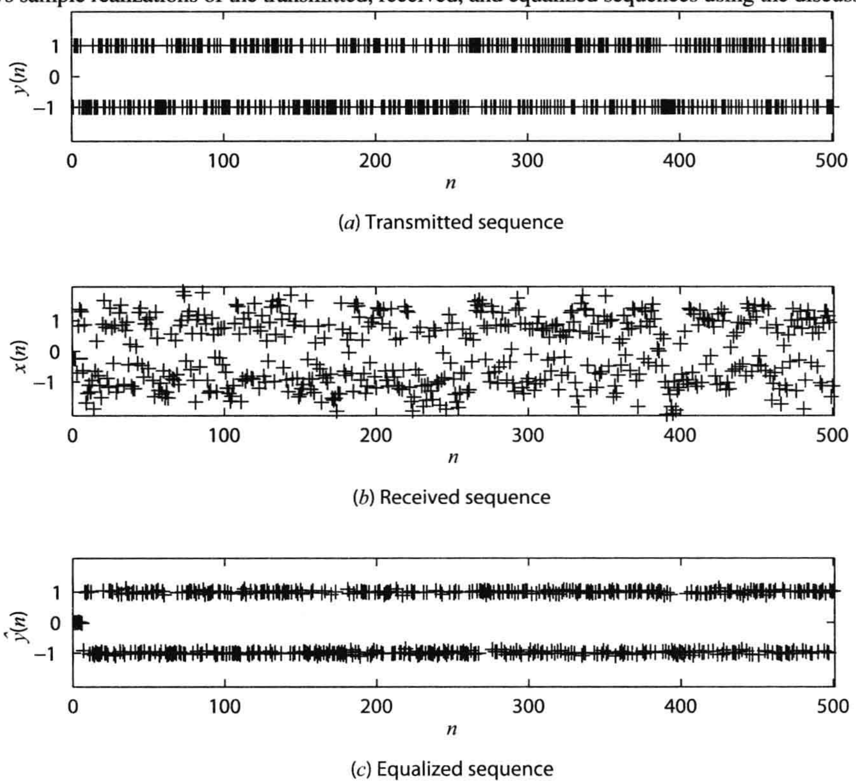


FIGURE 9.29
Sample realizations of the transmitted, received, and equalized sequences using an FIR LMS equalizer.

9.4.5 Some Practical Considerations

The LMS is the most widely known and used adaptive algorithm because of its simplicity and robustness to disturbances and model errors. We next discuss some issues related to its robustness, finite-word-length effects, and implementation.

Robustness

If we assume the model in Figure 9.19, an adaptive filter is said to be *robust* if the effect of the disturbances $\{\mathbf{c}(-1), e_o(n)\}$ on the resulting estimation errors $\{\tilde{\mathbf{c}}(n),\ e(n)\}$ (or $\{\tilde{\mathbf{c}}(n),\ \varepsilon(n)\}$), as measured by their energy, is small (Sayed and Rupp 1998). Basically a robust adaptive filter should be insensitive to the initial conditions $\mathbf{c}(-1)$ and the optimum residual error $e_o(n)$, which acts as measurement noise. These inputs are collectively called *disturbances*. In practice, $e_o(n)$ accounts not only for measurement noise but also for model mismatching, quantization errors, and other inaccuracies.

If we define the energies of the disturbances and the estimation errors by

$$E_{\text{dist}}(n) = \frac{1}{2\mu}\|\tilde{c}(-1)\|^2 + \sum_{j=0}^{n} |e_o(n)|^2 \tag{9.4.83}$$

and

$$E_{\text{error}}(n) = \frac{1}{2\mu}\|\tilde{c}(n)\|^2 + \sum_{j=0}^{n} |\tilde{y}(n)|^2 \tag{9.4.84}$$

it can be shown that the coefficient vectors determined by the LMS algorithm satisfy the condition

$$E_{\text{error}}(n) \le E_{\text{dist}}(n) \tag{9.4.85}$$

assuming that $0 < 2\mu \le 1/\|x(n)\|^2$ (Sayed and Kailath 1994; Sayed and Rupp 1996). Equation (9.4.85) shows that the energy of the residuals is always upper-bounded by the energy of the disturbances, which explains the robust behavior of the LMS algorithm.

Furthermore, it can be shown that the LMS algorithm minimizes the maximum possible difference between these two energies, over all disturbances with finite energy, and is optimum according to the H^∞ (or minimax) criterion (Sayed and Rupp 1998; Hassibi et al. 1996).

Finite-precision effects

When we design an LMS adaptive filter for a stationary SOE, we choose the step size μ to provide the desired balance between speed of convergence and misadjustment. If we are not concerned about fast convergence, we can reduce μ so much as to obtain practically insignificant misadjustment. However, in a digital implementation, the adaptation of the LMS algorithm stops (stalls) when the correction term becomes smaller in magnitude than one-half of the least significant bit (LSB), that is,

$$|2\mu e^*(n)x(n-k)| \le \frac{\text{LSB}}{2} \tag{9.4.86}$$

Therefore, a decrease in μ may result in a performance degradation, unless we increase the number of bits (i.e., the precision) of the filter coefficients. If X_{rms} is the root mean square (rms) amplitude of the input signal, to a good approximation we have

$$|e(n)| \le \frac{\text{LSB}}{4\mu X_{\text{rms}}} \triangleq \text{DRE} \tag{9.4.87}$$

where DRE is known as the *digital residual error* (Gitlin et al. 1973). We note that for a given number of bits the DRE increases as we decrease the step size μ.

The roundoff numerical errors contribute to the steady-state EMSE a term that is inversely proportional to μ, whereas the quantization of the input data and the filter output contributes a second term that is independent of the step size (Caraiscos and Liu 1984). Hence, in practice the step size of the LMS algorithm cannot be decreased below the level where the degradation effects of quantization and finite-precision arithmetic become significant. Also, the finite-precision effects become more pronounced as the ill conditioning of the input increases (Alexander 1987).

When one or more eigenvalues of the input correlation matrix are zero, the corresponding adaptation modes

either do not converge or may result in overflow due to nonlinear quantization effects (Gitlin et al. 1982). These effects can be prevented by using a technique known as *leakage*. The *leaky* LMS algorithm is given by

$$\boldsymbol{c}(n) = (1-\gamma\,\mu)\boldsymbol{c}(n-1) + \mu e^*(n)\boldsymbol{x}(n) \tag{9.4.88}$$

where γ is the leakage coefficient. Since μ and γ are very small positive constants, $1-\gamma\,\mu$ is slightly less than 1. The updating (9.4.88) is obtained by minimizing the cost function

$$P(n) = |e(n)|^2 + \gamma\|\boldsymbol{c}(n)\|^2 \tag{9.4.89}$$

which includes a penalty term proportional to the size of the coefficient vector. The price of leakage is an increase in computational complexity and some bias in the obtained estimates (see Problem 9.17). More details and practical applications of the leaky LMS algorithm to adaptive equalization are discussed in Gitlin et al. (1992, 1982).

We can simplify the hardware implementation of LMS adaptive filters by using nonlinearities to avoid the multiplications involved in the updating of the filter coefficients. These simplified LMS algorithms update the filter coefficients by using quantized correction terms such as $\mu\,\text{sign}\{e(n)\}x(n-k)$, $\mu e(n)\text{sign}\{x(n-k)\}$, or $\mu\,\text{sign}\{e(n)x(n-k)\}$; and their performance is degraded by the lower precision. Various signum-based LMS adaptive algorithms are discussed in Claasen and Mecklenbrauker (1981), Duttweiler (1982), and Treichler et al. (1987).

Transform-domain and block LMS algorithms

The LMS algorithm attains its best rate of convergence when the input correlation matrix is diagonal with equal eigenvalues. In the case of FIR filters, this implies that the input signal is white noise. When the components of the input data vector are correlated, we can improve the convergence by using an isotropic decorrelating transformation, as shown in Figure 9.30. The transformation matrix can be obtained by using either the triangular or the orthogonal decomposition of the input correlation matrix as explained in Section 2.3. Since the innovations vector used by the LMS algorithm has uncorrelated components with unit variance, the error performance surface is a hypersphere, and the *transform-domain LMS algorithm* attains its best rate of convergence. In practice, when the input correlation matrix is unknown and possibly time-varying, we can only use suboptimum transforms such as the DFT, the *discrete cosine transform* (*DCT*), the *discrete wavelet transform* (*DWT*), or some other orthogonal transform. The performance of the obtained adaptive filter depends on the decorrelation properties of the transform, which in turn depends on the properties of the input correlation matrix. Another approach to overcome the problem of slow convergence for highly correlated inputs is found in the family of *affine projection algorithms* discussed in Ozeki and Umeda (1984), Rupp (1995), and Morgan and Kratzer (1996) and the references therein.

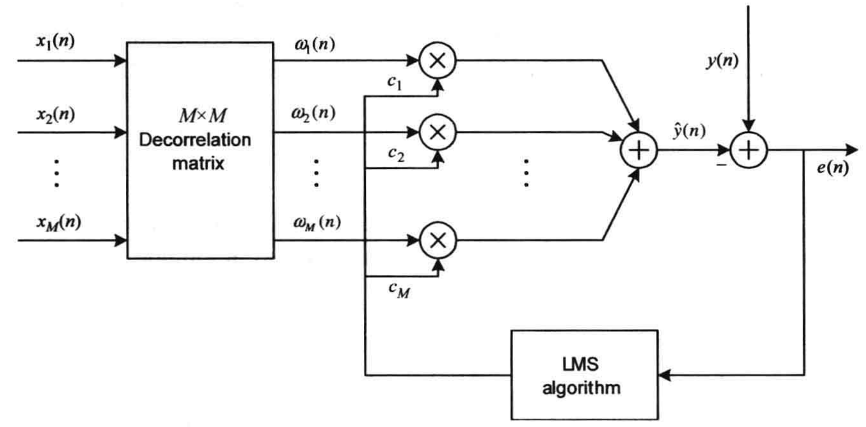


FIGURE 9.30
Transform domain LMS adaptive filter structure.

In applications that require adaptive filters with a very large number of coefficients, real-time implementation of the LMS algorithm becomes quite involved. For example, acoustic echo cancelers with 8000 coefficients (500 ms sampled at 16 kHz) are typical for teleconference applications (Gilloire et al. 1996). The complexity of such applications can be reduced by using block adaptive filters (see Figure 9.31) that process one block of data at a time in either the time or the frequency domain. The adaptive filter coefficients are updated once per block and are kept

fixed within the block. Such filters have good numerical accuracy, and can be easily pipelined and parallelized, and their complexity can be reduced by computing the involved convolutions and correlations using FFT algorithms. In some applications, such as acoustic echo cancelation, the block-length delay introduced by these filters may create problems. A detailed treatment of block and frequency-domain LMS algorithms is given in Shynk (1992), Gilloire et al. (1996), Haykin (1996), Jenkins and Marshall (1998), and Treichler et al. (1987).

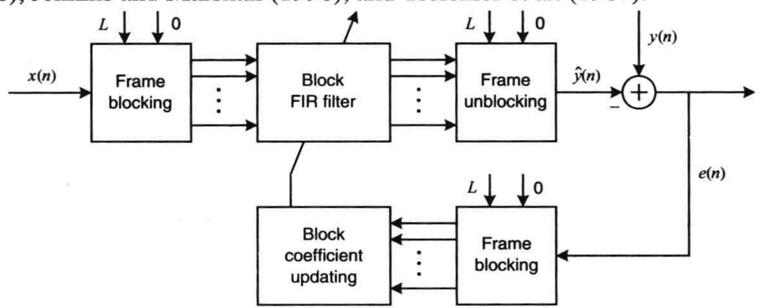


FIGURE 9.31
Block adaptive filter structure.

Another approach to reduce complexity and improve convergence is *subband adaptive filtering*, which splits the input signal and the desired response into smaller frequency bands (subbands), subsamples the resulting signals, processes each subband with different LMS filters, and finally interpolates and recombines the subbands to obtain the filter output (Shynk 1992; Gilloire and Vetterli 1992). The improved convergence results because the spectral dynamic range of each subband is smaller than that of the full band. However, the performance of subband adaptive filters is degraded by the cross-talk between adjacent subbands.

9.5 Recursive Least-Squares Adaptive Filters

In this section we use the method of LS to develop adaptive filters, we determine their rate of convergence and misadjustment, and we introduce the *conventional recursive least-squares (CRLS) algorithm* for their implementation. The CRLS algorithm does *not* impose any restrictions on the input data vector; therefore, it can be used for both array processing and FIR filtering applications.

9.5.1 LS Adaptive Filters

LS adaptive filters are designed so that the updating of their coefficients always attains the minimization of the total squared error from the time the filter initiated operation up to the current time. Therefore, the filter coefficients at time index n are chosen to minimize the cost function

$$E(n)=\sum_{j=0}^{n}\lambda^{n-j}\,|e(j)|^2=\sum_{j=0}^{n}\lambda^{n-j}\,|y(j)-\boldsymbol{c}^{\mathrm{H}}\boldsymbol{x}(j)|^2 \tag{9.5.1}$$

where $e(j)$ is the instantaneous error and the constant $\lambda,\ 0<\lambda\le 1$, is the *forgetting factor*. Note that since the filter coefficients are held constant during the observation interval $0\le j\le n$, then a priori and a posteriori errors are identical. The coefficient vector obtained by minimizing (9.5.1) is denoted by $\mathbf{c}(n)$ and provides the optimum LSE filter at time n. When $\lambda=1$, we say that the algorithm has *growing memory* because the values of the filter coefficients are a function of all the past input values. The forgetting factor (see Figure 9.32) is used to ensure that data in the distant past are paid less attention ("forgotten") in order to provide the filter with tracking capability when it operates in a varying SOE (see Section 9.8).

The filter coefficients that minimize the total squared error (9.5.1) are specified by the normal equations

$$\hat{\boldsymbol{R}}(n)\boldsymbol{c}(n)=\hat{\boldsymbol{d}}(n) \tag{9.5.2}$$

where

$$\hat{\boldsymbol{R}}(n)\triangleq\sum_{j=0}^{n}\lambda^{n-j}\,\boldsymbol{x}(j)\boldsymbol{x}^{\mathrm{H}}(j) \tag{9.5.3}$$

and

$$\hat{\boldsymbol{d}}(n)\triangleq\sum_{j=0}^{n}\lambda^{n-j}\,\boldsymbol{x}(j)y^{*}(j) \tag{9.5.4}$$

FIGURE 9.32
Exponential weighting of observations at times n and n+1. Older data are more heavily discounted by the algorithm.

provide exponentially weighted estimates of the input correlation matrix and the crosscorrelation vectorvetween input and desired response due to the presence of λ^{n-j} in the cost function (9.5.1). The minimum total squared error is

$$E_{\min}(n) = E_y(n) - \hat{\boldsymbol{d}}^{\mathrm{H}}(n)\boldsymbol{c}(n) \tag{9.5.5}$$

where E

$$E_y(n) \triangleq \sum_{j=0}^{n} \lambda^{n-j} \,|\, y(j)\,|^2 \tag{9.5.6}$$

is the energy of the weighted desired response signal. These formulas have been derived in Section 7.2.1.

Suppose now that we wait for some $n > M$, where $\hat{\boldsymbol{R}}(n)$ is usually nonsingular, we compute $\hat{\boldsymbol{R}}(n)$ and $\hat{\boldsymbol{d}}(n)$, and then we solve the normal equations (9.5.2) to determine the filter coefficients $\boldsymbol{c}(n)$. This approach, which is time-consuming, should be repeated with the arrival of new pairs of observations $\{\boldsymbol{x}(n),\ y(n)\}$, that is, at times $n+1,\ n+2,$ etc .

A first reduction in computational complexity can be obtained by noticing that (9.5.3) can be expressed as

$$\hat{\boldsymbol{R}}(n) = \lambda\hat{\boldsymbol{R}}(n-1) + \boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n) \tag{9.5.7}$$

which shows that the "new" correlation matrix $\hat{\boldsymbol{R}}(n)$ can be updated by weighting the "old" correlation matrix $\hat{\boldsymbol{R}}(n-1)$ with the forgetting factor λ and then incorporating the "new information" $\boldsymbol{x}(n)\boldsymbol{x}^H(n)$. Since the outer product $\boldsymbol{x}(n)\boldsymbol{x}^H(n)$ is a matrix of rank 1, (9.5.7) provides a rank 1 modification of the correlation matrix. Similarly, using (9.5.4), we can show that

$$\hat{\boldsymbol{d}}(n) = \lambda\hat{\boldsymbol{d}}(n-1) + \boldsymbol{x}(n)y^*(n) \tag{9.5.8}$$

which provides a time update of the cross-correlation vector.

We next show that using these two updatings, we can determine the new coefficient vector $\boldsymbol{c}(n)$ from the old coefficient vector $\boldsymbol{c}(n-1)$ and the new observation pair $\{\boldsymbol{x}(n), y(n)\}$ without solving the normal equations (9.5.2) from scratch.

A priori adaptive LS algorithm. If we solve (9.5.7) for $\hat{\boldsymbol{R}}(n-1)$ and (9.5.8) for $\hat{\boldsymbol{d}}(n-1)$ and use the normal equations (9.5.2), we have

$$[\hat{\boldsymbol{R}}(n) - \boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)]\boldsymbol{c}(n-1) = \hat{\boldsymbol{d}}(n) - \boldsymbol{x}(n)y^*(n)$$

or after some simple manipulations

$$\hat{\boldsymbol{R}}(n)\boldsymbol{c}(n-1) + \boldsymbol{x}(n)e^*(n) = \hat{\boldsymbol{d}}(n) \tag{9.5.9}$$

where

$$e(n) = y(n) - \boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n) \tag{9.5.10}$$

is the a priori estimation error. If the matrix $\hat{\boldsymbol{R}}(n)$ is invertible, by multiplying both sides of (9.5.9) by $\hat{\boldsymbol{R}}^{-1}(n)$ and using (9.5.2), we obtain

$$\boldsymbol{c}(n-1) + \hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)e^*(n) = \hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{d}}(n) = \boldsymbol{c}(n) \tag{9.5.11}$$

If we define the *adaptation gain vector* $\boldsymbol{g}(n)$ by

$$\hat{\boldsymbol{R}}(n)\boldsymbol{g}(n) \triangleq \boldsymbol{x}(n) \tag{9.5.12}$$

Equation (9.5.11) can be written as

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \boldsymbol{g}(n)e^*(n) \tag{9.5.13}$$

which shows how to update the old coefficient vector $\boldsymbol{c}(n-1)$ to obtain the current vector $\boldsymbol{c}(n)$.

EXAMPLE 9.5.1. It is instructive at this point to derive the LS adaptive filter with a single coefficient. Indeed, since for $M = 1$ the correlation matrix $\hat{\boldsymbol{R}}(n)$ becomes the scalar $E_x(n)$, we obtain

$$E_x(n) = \lambda E_x(n-1) + |x(n)|^2$$
$$e(n) = y(n) - c^*(n-1)x(n)$$
$$c(n) = c(n-1) + \frac{1}{E_x(n)}x(n)e^*(n)$$

which is like an LMS algorithm with time-varying gain $\mu(n) = 1/E_x(n)$. However, the present algorithm is optimum in the LS sense.

A posteriori adaptive LS algorithm. If we substitute (9.5.7) and (9.5.8) into the normal equations (9.5.2), after some simple manipulations, we obtain

$$\lambda\hat{\boldsymbol{R}}(n-1)\boldsymbol{c}(n) - \boldsymbol{x}(n)\varepsilon^*(n) = \lambda\hat{\boldsymbol{d}}(n-1) \tag{9.5.14}$$

where
$$\varepsilon(n) = y(n) - \boldsymbol{c}^{\mathrm{H}}(n)\boldsymbol{x}(n) \tag{9.5.15}$$
is the a posteriori estimation error. If the matrix $\hat{\boldsymbol{R}}(n-1)$ is invertible, (9.5.14) gives

$$\boldsymbol{c}(n) - \lambda^{-1}\hat{\boldsymbol{R}}^{-1}(n-1)\boldsymbol{x}(n)\varepsilon^*(n) = \hat{\boldsymbol{R}}^{-1}(n-1)\hat{\boldsymbol{d}}(n-1) = \boldsymbol{c}(n-1)$$

or
$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \bar{\boldsymbol{g}}(n)\varepsilon^*(n) \tag{9.5.16}$$

or
$$\lambda\hat{\boldsymbol{R}}(n-1)\bar{\boldsymbol{g}}(n) \triangleq \boldsymbol{x}(n) \tag{9.5.17}$$

determines the *alternative adaptation gain vector* $\bar{\boldsymbol{g}}(n)$.

Since recursions (9.5.15) and (9.5.16) are coupled, the a posteriori algorithm is not applicable. However, if we substitute (9.5.16) into (9.5.15), we obtain

$$\varepsilon(n) = y(n) - [\boldsymbol{c}^{\mathrm{H}}(n-1) + \varepsilon(n)\bar{\boldsymbol{g}}^{\mathrm{H}}(n)]\boldsymbol{x}(n)$$
$$= e(n) - \varepsilon(n)\bar{\boldsymbol{g}}^{\mathrm{H}}(n)\boldsymbol{x}(n)$$

or
$$\varepsilon(n) = \frac{e(n)}{\bar{\alpha}(n)} \tag{9.5.18}$$

where
$$\bar{\alpha}(n) \triangleq 1 + \bar{\boldsymbol{g}}^{\mathrm{H}}(n)\boldsymbol{x}(n) = 1 + \lambda^{-1}\boldsymbol{x}^{\mathrm{H}}(n)\hat{\boldsymbol{R}}^{-1}(n-1)\boldsymbol{x}(n) \tag{9.5.19}$$

is known as the *conversion factor*. Hence, we can use (9.5.19) and (9.5.18) to compute the a posteriori error $\varepsilon(n)$ before we update the filter coefficient vector. This trick makes possible the realization and use of the a posteriori LS adaptive filter algorithm. If $\hat{\boldsymbol{R}}(n-1)$ is positive definite, we have $\bar{\alpha}(n) > 1$ and $|\varepsilon(n)| < |e(n)|$ for all n. Therefore,

$$\sum_n |\varepsilon(n)|^2 < \sum_n |e(n)|^2 \tag{9.5.20}$$

which should be expected[①] because the adaptive filter is designed by minimizing, at each time n, the total squared a posteriori error $\varepsilon(n)$.

Also, from (9.5.13), (9.5.16), and (9.5.18) we obtain

$$\mathbf{g}(n) = \frac{\bar{\mathbf{g}}(n)}{\bar{\alpha}(n)} \tag{9.5.21}$$

which shows that the two adaptation gains have the same direction but different lengths. However, from (9.5.13) and (9.5.16) we see that the corrections $\mathbf{g}(n)e^*(n)$ and $\bar{\mathbf{g}}(n)\varepsilon^*(n)$ are equal.

Another conversion factor, defined in terms of the gain vector $\mathbf{g}(n)$, is

$$\alpha(n) \triangleq 1 - \mathbf{x}^{\mathrm{H}}(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) = 1 - \mathbf{x}^{\mathrm{H}}(n)\mathbf{g}(n) \tag{9.5.22}$$

and has some interesting interpretations. Using (9.5.21), we have

$$\alpha(n) = 1 - \frac{\mathbf{x}^{\mathrm{H}}(n)\bar{\mathbf{g}}(n)}{\bar{\alpha}(n)}$$

$$\alpha(n)\bar{\alpha}(n) = \bar{\alpha}(n) + 1 - [1 + \mathbf{x}^{\mathrm{H}}(n)\bar{\mathbf{g}}(n)] = 1$$

or

$$\alpha(n) = \frac{1}{\bar{\alpha}(n)} \tag{9.5.23}$$

which shows that the two conversion factors are inverses of each other. Since the input correlation matrix is nonnegative definite, that is, $\mathbf{x}^{\mathrm{H}}(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) \geq 0$, (9.5.22) implies

$$0 < \alpha(n) \leq 1 \tag{9.5.24}$$

that is, the conversion factor $\alpha(n)$ is bounded by 0 and 1. This bound allows the interpretation of $\alpha(n)$ as an angle variable (Lee et al. 1981), and its monitoring can provide information about the proper operation of RLS algorithms. Also the quantity $1-\alpha(n)$ can be interpreted as a *likelihood variable* (Lee et al. 1981). It can be shown (see Problem 9.23) that

$$\alpha(n) = \lambda^M \frac{\det \hat{\mathbf{R}}(n-1)}{\det \hat{\mathbf{R}}(n)} \tag{9.5.25}$$

which shows the importance of $\alpha(n)$ or $\bar{\alpha}(n)$ for the invertibility for the estimated correlation matrix.

The computational organization of the a priori and a posteriori LS adaptive algorithms is summarized in Table 9.5.

TABLE 9.5

Summary of a priori and a posteriori LS adaptive filter approaches.

	A priori LS adaptive filter	A posteriori LS adaptive filter
Correlation matrix	$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)$	$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)$
Adaptation gain	$\hat{\mathbf{R}}(n)\mathbf{g}(n) = \mathbf{x}(n)$	$\lambda\hat{\mathbf{R}}(n-1)\bar{\mathbf{g}}(n) = \mathbf{x}(n)$
A priori error	$e(n) = y(n) - \mathbf{c}^{\mathrm{H}}(n-1)\mathbf{x}(n)$	$e(n) = y(n) - \mathbf{c}^{\mathrm{H}}(n-1)\mathbf{x}(n)$
Conversion factor	$\alpha(n) = 1 - \mathbf{g}^{\mathrm{H}}(n)\mathbf{x}(n)$	$\bar{\alpha}(n) = 1 + \bar{\mathbf{g}}^{\mathrm{H}}(n)\mathbf{x}(n)$
A posteriori error	$\varepsilon(n) = \alpha(n)e(n)$	$\varepsilon(n) = \dfrac{e(n)}{\bar{\alpha}(n)}$
Coefficient updating	$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$	$\mathbf{c}(n) = \mathbf{c}(n-1) + \bar{\mathbf{g}}(n)\varepsilon^*(n)$

[①] The computation of the quantity $\sum_{j=0}^{n}\lambda^{n-j}|y(j) - \mathbf{c}^{\mathrm{H}}\mathbf{x}(j)|^2$ for $\mathbf{c} = \mathbf{c}(n), \mathbf{c}(j)$, or $\mathbf{c}(j-1)$ gives the block, a posteriori, or a priori total squared error. Clearly, only the block filter performs optimum LS filtering for all data in the interval $0 \leq j \leq n$ (see Problem 10.22).

Figure 9.33 shows a block diagram representation of the a priori LS adaptive filter. There are two important points to be made:

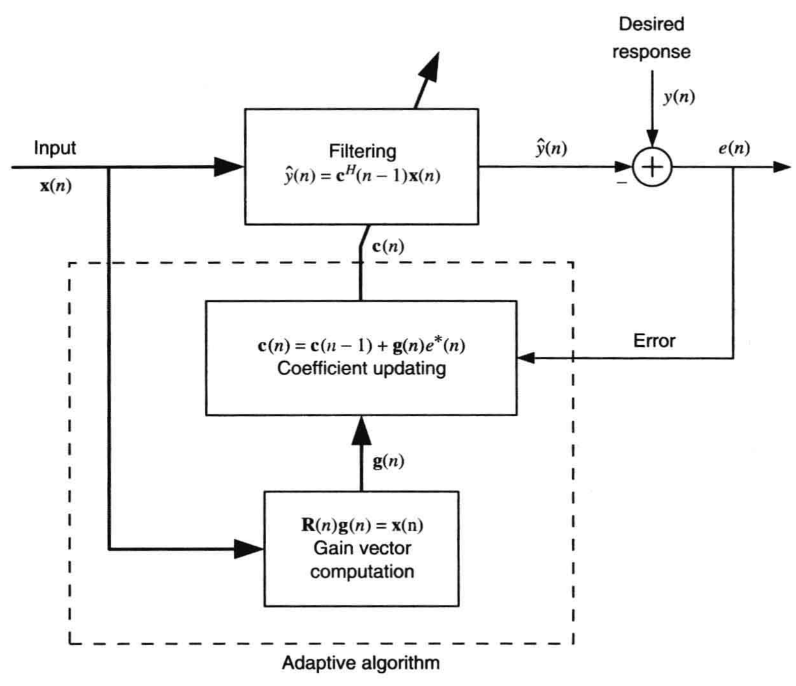


FIGURE 9.33
Basic elements of the a priori LS adaptive filter. Note that the filtering process has no effect on the computation of the gain vector.

- The adaptation gain is strictly a function of the input signal. The desired response only affects the magnitude and sign of the coefficient correction term through the error.
- The most demanding computational task in RLS filtering is the computation of the adaptation gain. This involves the solution of a linear system of equations, which requires $O(M^3)$ operations per time update.

9.5.2 Conventional Recursive Least-Squares Algorithm

The major computational load in LS adaptive filters, that is, the computation of the gain vectors

$$g(n) = \hat{R}^{-1}(n)x(n) \tag{9.5.26}$$

or

$$\bar{g}(n) = \lambda^{-1}\hat{R}^{-1}(n-1)x(n) \tag{9.5.27}$$

can be reduced if we can find a recursive formula to update the inverse

$$P(n) \triangleq \hat{R}^{-1}(n) \tag{9.5.28}$$

of the correlation matrix. We can develop such an updating by using the rank 1 updating (9.5.7) and the matrix inversion lemma

$$(\lambda R + xx^{\mathrm{H}})^{-1} = \lambda^{-1}R^{-1} - \frac{(\lambda^{-1}R^{-1}x)(\lambda^{-1}R^{-1}x)^{\mathrm{H}}}{1+\lambda^{-1}x^{\mathrm{H}}R^{-1}x} \tag{9.5.29}$$

discussed in Appendix A.

Indeed, using (9.5.29), (9.5.7), (9.5.26), and (9.5.19), we can easily show that

$$P(n) = \lambda^{-1}P(n-1) - g(n)\bar{g}^{\mathrm{H}}(n) \tag{9.5.30}$$

which provides the desired updating formula. Indeed, given the old matrix $\boldsymbol{P}(n-1)$ and the new observations $\{\boldsymbol{x}(n),\ y(n)\}$ we compute the new matrix $\boldsymbol{P}(n)$, using the following procedure

$$\begin{aligned}\bar{\boldsymbol{g}}(n) &= \lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{x}(n)\\ \alpha(n) &= 1+\bar{\boldsymbol{g}}^{\mathrm{H}}(n)\boldsymbol{x}(n)\\ \boldsymbol{g}(n) &= \frac{\bar{\boldsymbol{g}}(n)}{\bar{\alpha}(n)}\\ \boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1)-\boldsymbol{g}(n)\bar{\boldsymbol{g}}^{\mathrm{H}}(n)\end{aligned} \tag{9.5.31}$$

which is known as the *conventional recursive LS (CRLS) algorithm*. We again stress that *the CRLS algorithm is valid for both linear combiners and FIR filters because it does not make any assumptions about the nature of the input data vector*. However, for FIR filters we usually assume prewindowing, that is, $\boldsymbol{x}(-1)=\mathbf{0}$, or equivalently $x(n)=0$ for $-M \le n \le -1$.

Updating of the minimum total squared error. We next derive an update recursion for the minimum total squared error (9.5.5). Using (9.5.6), we can easily see that

$$E_y(n)=\lambda E_y(n-1)+y(n)y^*(n) \tag{9.5.32}$$

which provides a recursive updating for the energy of the desired response. Substituting (9.5.32) and (9.5.13) into (9.5.5), we obtain

$$E_{\min}(n)=\lambda E_y(n-1)+y(n)y^*(n)-\hat{\boldsymbol{d}}^{\mathrm{H}}(n)\boldsymbol{c}(n-1)-\hat{\boldsymbol{d}}^{\mathrm{H}}(n)\boldsymbol{g}(n)e^*(n)$$

or by using (9.5.8)

$$\begin{aligned}E_{\min}(n)=\lambda E_y(n-1)&+y(n)y^*(n)-\hat{\boldsymbol{d}}^{\mathrm{H}}(n)\boldsymbol{g}(n)e^*(n)\\ &-y(n)\boldsymbol{x}^{\mathrm{H}}(n)\boldsymbol{c}(n-1)-\lambda\hat{\boldsymbol{d}}^{\mathrm{H}}(n-1)\boldsymbol{c}(n-1)\end{aligned}$$

Rearranging the terms of the last equation and using (9.5.5), we have

$$\begin{aligned}E_{\min}(n)&=\lambda[E_y(n-1)-\hat{\boldsymbol{d}}^{\mathrm{H}}(n-1)\boldsymbol{c}(n-1)]+[y(n)-\hat{\boldsymbol{d}}^{\mathrm{H}}(n)\boldsymbol{g}(n)]e^*(n)\\ &=\lambda E_{\min}(n-1)+\{y(n)-[\hat{\boldsymbol{d}}^{\mathrm{H}}(n)\hat{\boldsymbol{R}}^{-1}(n)][\hat{\boldsymbol{R}}(n)\boldsymbol{g}(n)]\}e^*(n)\\ &=\lambda E_{\min}(n-1)+[y(n)-\boldsymbol{c}^{\mathrm{H}}(n)\boldsymbol{x}(n)]e^*(n)\end{aligned}$$

where the last equation is obtained because the matrix $\hat{\boldsymbol{R}}(n)$ and its inverse are Hermitian. The last equation leads to

$$E_{\min}(n)=\lambda E_{\min}(n-1)+\varepsilon(n)e^*(n) \tag{9.5.33}$$

$$=\lambda E_{\min}(n-1)+\bar{\alpha}(n)\,|\,\varepsilon(n)\,|^2 \tag{9.5.34}$$

$$=\lambda E_{\min}(n-1)+\frac{|\,e(n)\,|^2}{\alpha(n)} \tag{9.5.35}$$

which provide the desired updating formulas. Since the product $\varepsilon(n)e^*(n)$ is by necessity real, we have $\varepsilon(n)e^*(n)=\varepsilon^*(n)e(n)$. The value of $E_{\min}(n)$ increases with time and reaches a finite limit value only if $\lambda<1$.

9.5.3 Some Practical Considerations

In the practical implementation of CRLS adaptive filters, we have to deal with the issues of computational complexity, initialization, and finite-word-length effects.

Computational complexity. The complete CRLS algorithm is summarized in Table 9.6. A measure of the

computational complexity of the CRLS algorithm is provided by the number of operations (one operation consists of one multiplication and one addition) required to perform one updating. Since $\boldsymbol{P}(n)$ is Hermitian, it is possible to implement the algorithm so that it will require $2M^2+4M$ operations per time updating. The computation of $\bar{\boldsymbol{g}}_\lambda(n)$ and the updating of $\boldsymbol{P}(n)$ require $O(M^2)$ operations. In contrast, all remaining formulas, which involve dot products and vector-by-scalar multiplications, require $O(M)$ operations. The inversion of the correlation matrix $\hat{\boldsymbol{R}}(n)$ is essentially replaced by the scalar division used to compute $\mathbf{g}(n)$.

TABLE 9.6
Practical implementation of the RLS algorithm. To update $\mathrm{P}(n)$, we only compute its upper (low) triangular part and determine the other part using Hermitian symmetry.

Initialization
$\boldsymbol{c}(-1)=0 \quad \boldsymbol{P}(-1)=\delta^{-1}\boldsymbol{I}$ $\delta=$ small positive constant
For each $n=0,1,2,\cdots$ compute:
Adaptation gain computation
$\bar{\boldsymbol{g}}_\lambda(n)=\boldsymbol{P}(n-1)\boldsymbol{x}(n) \quad \alpha_\lambda(n)=\lambda+\bar{\boldsymbol{g}}_\lambda^{\mathrm{H}}(n)\boldsymbol{x}(n)$ $\boldsymbol{g}(n)=\dfrac{\bar{\boldsymbol{g}}_\lambda(n)}{\alpha_\lambda(n)} \quad \boldsymbol{P}(n)=\lambda^{-1}[\boldsymbol{P}(n-1)-\boldsymbol{g}(n)\bar{\boldsymbol{g}}_\lambda^{\mathrm{H}}(n)]$
Filtering
$e(n)=y(n)-\boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)$
Coefficient updating
$\boldsymbol{c}(n)=\boldsymbol{c}(n-1)+\boldsymbol{g}(n)e^*(n)$

Initialization. There are two ways to obtain the values $\boldsymbol{P}(-1)$ and $\mathbf{c}(-1)$ required to initialize the CRLS algorithm. The most obvious way is to collect an initial block of data $\{\mathbf{x}(n),\ y(n)\}_{-n_0}^{-1}, n_0>M$, and then compute the exact inverse matrix $\boldsymbol{P}(-1)$ and the exact LS solution $\boldsymbol{c}(-1)$.

The approach used in practice is to set $\boldsymbol{P}(-1)=\delta^{-1}\boldsymbol{I}$, where δ is a very small positive number (on the order of $0.01\sigma_x^2$) and $\boldsymbol{c}(-1)=0$. For FIR filters this corresponds to setting $x(-M+1)=\sqrt{\delta}$ and $x(n)=0$ for $-M+2 \le n \le -1$. For any $n>M$, the normal equations matrix is $\delta\lambda^n\boldsymbol{I}+\hat{\boldsymbol{R}}(n)$ and results in a biased estimate of $\boldsymbol{c}(n)$. However, for large *n* the choice of δ is unimportant because the algorithm has exponentially forgetting memory for $\lambda<1$.

It can be shown (see Problem 9.24) that this approach provides a set of coefficients that minimizes the modified cost function

$$E(n)=\delta\lambda^{n+1}\|\boldsymbol{c}\|^2+\sum_{j=0}^{n}\lambda^{n-j}\,|\,y(j)-\boldsymbol{c}^H\boldsymbol{x}(j)\,|^2 \tag{9.5.36}$$

instead of (9.5.1). Note that if we turn off the input, that is, we set $\boldsymbol{x}(n)=\mathbf{0}$, then (9.5.30) becomes $\boldsymbol{P}(n)=\lambda^{-1}\boldsymbol{P}(n-1)$, which is an unstable recursion when $\lambda<1$.

Finite-word-length effects. There are different RLS algorithms that are algebraically equivalent; that is, they solve the same set of normal equations. Therefore, they have the same rate of convergence and the same insensitivity to variations in the eigenvalue spread of the input correlation matrix with the CRLS algorithm. All RLS algorithms are obtained by exploiting exact mathematical relations between various algorithmic quantities to obtain better computational or numerical properties. Many of these algorithmic quantities have certain physical meanings or theoretical properties. For example, in the CRLS algorithm, the matrix $\boldsymbol{P}(n)$ is Hermitian and positive definite, the angle variable satisfies $0<\alpha(n)\le 1$, and energy $E(n)$ should be always positive. However, when we use finite precision, some of these exact relations, properties, or acceptable ranges for certain algorithmic variables may be violated.

The numerical instability of RLS algorithms can be traced to such forms of *numerical inconsistencies* (Verhaegen 1989; Yang and Böhme 1992; Haykin 1996). The crucial part of the CRLS algorithm is the updating of the inverse correlation matrix $\boldsymbol{P}(n)$ via (9.5.30). The CRLS algorithm becomes numerically unstable when the matrix $\boldsymbol{P}(n)=\hat{\boldsymbol{R}}^{-1}(n)$ loses its Hermitian symmetry or its positive definiteness (Verhaegen 1989). In practice, we can preserve the Hermitian symmetry of $\boldsymbol{P}(n)$ by computing only its lower (or upper) triangular part, using (9.5.30), and then filling the other part, using the relation $p_{ij}(n)=p_{ji}^*(n)$. Another approach is to replace $\boldsymbol{P}(n)$ by $[\boldsymbol{P}(n)+\boldsymbol{P}^{\mathrm{H}}(n)]/2$ after updating from $\boldsymbol{P}(n-1)$ to $\boldsymbol{P}(n)$.

It has been shown that the CRLS algorithm is numerically stable for $\lambda<1$ and diverges for $\lambda=1$ (Ljung and Ljung 1985).

9.5.4 Convergence and Performance Analysis

The purpose of any LS adaptive filter, in a stationary SOE, is to identify the optimum filter $\boldsymbol{c}_o=\boldsymbol{R}^{-1}\boldsymbol{d}$ from observations of the input vector $\boldsymbol{x}(n)$ and the desired response

$$y(n)=\boldsymbol{c}_o^{\mathrm{H}}\boldsymbol{x}(n)+e_o(n) \tag{9.5.37}$$

To simplify the analysis we adopt the independence assumptions discussed in Section 9.4.2. The results of the subsequent analysis hold for any LS adaptive filter implemented using the CRLS method or any other algebraically equivalent algorithm. We derive separate results for the growing memory and the fading memory (exponential forgetting) algorithms.

Growing memory $(\lambda=1)$

In this case all the values of the error signal, from the time the filter starts its operation to the present, have the same influence on the cost function. As a result, the filter loses its tracking ability, which is not important if the filter is used in a stationary SOE.

Convergence in the mean. For $n>M$ the coefficient vector $\boldsymbol{c}(n)$ is identical to the block LS solution discussed in Section 7.2.2. Therefore

$$E\{\boldsymbol{c}(n)\}=\boldsymbol{c}_o \qquad \text{for } n>M \tag{9.5.38}$$

that is, the RLS algorithm converges in the mean for $n>M$, where M is the number of coefficients.

Mean square deviation. For $n>M$ we have

$$\boldsymbol{\Phi}(n)=\sigma_o^2 E\{\hat{\boldsymbol{R}}^{-1}(n)\} \tag{9.5.39}$$

because $\boldsymbol{c}(n)$ is an exact LS estimate (see Section 7.2.2). The correlation matrix $\hat{\boldsymbol{R}}(n)$ is described by a complex Wishart distribution, and the expectation of its inverse is

$$E\{\hat{\boldsymbol{R}}^{-1}(n)\}=\frac{1}{n-M}\boldsymbol{R}^{-1} \qquad n>M \tag{9.5.40}$$

as shown in Muirhead (1982) and Haykin (1996). Hence

$$\boldsymbol{\Phi}(n)=\frac{\sigma_o^2}{n-M}\boldsymbol{R}^{-1} \qquad n>M \tag{9.5.41}$$

and the MSD is

$$D(n)=\operatorname{tr}[\boldsymbol{\Phi}(n)]=\frac{\sigma_o^2}{n-M}\sum_{i=1}^{M}\frac{1}{\lambda_i} \qquad n>M \tag{9.5.42}$$

where λ_i, the eigenvalues of $\boldsymbol{R}$, should not be confused with the forgetting factor λ. From (9.5.42) we conclude that (1) the MSD is magnified by the smallest eigenvalue of $\boldsymbol{R}$ and (2) the MSD decays almost linearly with time.

A priori excess MSE. We now focus on the a priori LS algorithm because it is widely used in practice and to

facilitate a fairer comparison with the (a priori) LMS algorithm. To this end, we note that the a priori excess MSE formula (9.4.48)

$$P_{ex}(n) = \text{tr}[\boldsymbol{R\Phi}(n-1)] \tag{9.5.43}$$

derived in Section 9.4.2, under the independence assumption, holds for any a priori adaptive algorithm. Hence, substituting (9.5.41) into (9.5.43), we obtain

$$P_{ex}(n) = \frac{M}{n-M-1}\sigma_o^2 \qquad n > M \tag{9.5.44}$$

which shows that $P_{ex}(n)$ tends to zero as $n \to \infty$.

Exponentially decaying memory $(0 < \lambda < 1)$

In this case the most recent values of the observations have greater influence on the formation of the LS estimate of the filter coefficients. The memory of the filter, that is, the *effective* number of samples used to form the various estimates, is about $1/(1-\lambda)$ for $0.95 < \lambda < 1$ (see Section 9.8).

Convergence in the mean. We start by multipying both sides of (9.5.11) by $\hat{\boldsymbol{R}}(n)$, and then we use (9.5.7) and (9.5.10) to obtain

$$\hat{\boldsymbol{R}}(n)\boldsymbol{c}(n) = \lambda\hat{\boldsymbol{R}}(n-1)\boldsymbol{c}(n-1) + \boldsymbol{x}(n)y^*(n) \tag{9.5.45}$$

If we multiply (9.5.7) by $\boldsymbol{c}_o$ and subtract the resulting equation from (9.5.45), we get

$$\hat{\boldsymbol{R}}(n)\tilde{\boldsymbol{c}}(n) = \lambda\hat{\boldsymbol{R}}(n-1)\tilde{\boldsymbol{c}}(n-1) + \boldsymbol{x}(n)e_o^*(n) \tag{9.5.46}$$

where $\tilde{\boldsymbol{c}}(n) = \boldsymbol{c}(n) - \boldsymbol{c}_o$ is the coefficient error vector. Solving (9.5.46) by recursion, we obtain

$$\tilde{\boldsymbol{c}}(n) = \lambda^n \hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{R}}(0)\tilde{\boldsymbol{c}}(0) + \hat{\boldsymbol{R}}^{-1}(n)\sum_{j=0}^{n}\lambda^{n-j}\boldsymbol{x}(j)e_o^*(j) \tag{9.5.47}$$

which depends on the initial conditions and the optimum error $e_o(n)$. If we assume that $\hat{\boldsymbol{R}}(n)$, $\boldsymbol{x}(j)$, and $e_o(j)$ are independent and we take the expectation of (9.5.47), we obtain

$$E\{\tilde{\boldsymbol{c}}(n)\} = \delta\lambda^n E\{\hat{\boldsymbol{R}}^{-1}(n)\}\tilde{\boldsymbol{c}}(0) \tag{9.5.48}$$

where, as usual, we have set $\hat{\boldsymbol{R}}(0) = \delta\boldsymbol{I}$, $\delta > 0$. If the matrix $\hat{\boldsymbol{R}}(n)$ is positive definite and $0 < \lambda < 1$, then the mean vector $E\{\tilde{\boldsymbol{c}}(n)\} \to 0$ as $n \to \infty$. Hence, the RLS algorithm with exponential forgetting converges asymptotically in the mean to the optimum filter.

Mean square deviation. Using (9.5.46), we obtain the following difference equation for the coefficient error vector

$$\tilde{\boldsymbol{c}}(n) = \lambda\hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{R}}(n-1)\tilde{\boldsymbol{c}}(n-1) + \hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)e_o^*(n)$$

or

$$\tilde{\boldsymbol{c}}(n) \simeq \lambda\tilde{\boldsymbol{c}}(n-1) + \hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)e_o^*(n)$$

because $\hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{R}}(n-1) \simeq \boldsymbol{I}$ for large n. If we neglect the dependence among $\tilde{\boldsymbol{c}}(n-1)$, $\hat{\boldsymbol{R}}(n)$, $\boldsymbol{x}(n)$, and $e_o(n)$, we have

$$\boldsymbol{\Phi}(n) \simeq \lambda^2\boldsymbol{\Phi}(n-1) + \sigma_o^2 E\{\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\hat{\boldsymbol{R}}^{-1}(n)\} \tag{9.5.49}$$

where $\sigma_o^2 = E\{|e_o(n)|^2\}$.

To make the analysis mathematically tractable, we need an approximation for the inverse matrix $\hat{\boldsymbol{R}}^{-1}(n)$. To this end, using (9.5.3), we have

$$E\{\hat{\boldsymbol{R}}(n)\} = \sum_{j=0}^{n}\lambda^{n-j}E\{\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{H}}(n)\} = \frac{1-\lambda^{n+1}}{1-\lambda}\boldsymbol{R} \simeq \frac{1}{1-\lambda}\boldsymbol{R} \tag{9.5.50}$$

where the last approximation holds for $n \gg 1$. If we use the approximation $E\{\hat{\boldsymbol{R}}(n)\} \simeq \hat{\boldsymbol{R}}(n)$, we obtain

$$\hat{R}^{-1}(n) \simeq (1-\lambda)R^{-1} \tag{9.5.51}$$

which is more rigorously justified in Eleftheriou and Falconer (1986). Using the last approximation, (9.5.50) becomes

$$\Phi(n) \simeq \lambda^2 \Phi(n-1) + (1-\lambda)^2 \sigma_o^2 R^{-1} \tag{9.5.52}$$

which converges because $\lambda^2 < 1$. At steady state we have

$$(1-\lambda^2)\Phi(\infty) \simeq (1-\lambda)^2 \sigma_o^2 R^{-1}$$

because $\Phi(n) \simeq \Phi(n-1)$ for $n \gg 1$. Hence

$$\Phi(\infty) \simeq \frac{1-\lambda}{1+\lambda} \sigma_o^2 R^{-1} \tag{9.5.53}$$

and therefore

$$\mathcal{D}_\lambda(\infty) = \text{tr}[\Phi(\infty)] = \frac{1-\lambda}{1+\lambda} \sigma_o^2 \sum_{i=1}^{M} \frac{1}{\lambda_i} \tag{9.5.54}$$

which in contrast to (9.5.42) does not converge to zero as $n \to \infty$. This is explained by noticing that when $\lambda < 1$, the RLS algorithm has finite memory and does not use effectively all the data to form its estimate.

Steady-state a priori excess MSE. From (9.5.43) and (9.5.53) we obtain

$$P_{\text{ex}}(\infty) = \text{tr}[R\Phi(\infty)] \simeq \frac{1-\lambda}{1+\lambda} M \sigma_o^2 \tag{9.5.55}$$

which shows that as a result of finite memory, there is a steady-state excess MSE that decreases as λ approaches 1, that is, as the effective memory of the algorithm increases.

Summary

The results of the above analysis are summarized in Table 9.7 for easy reference. We stress at this point that all RLS algorithms, independent of their implementation, have the same performance, assuming that we use sufficient numerical precision (e.g., double-precision floating-point arithmetic). Sometimes, RLS algorithms are said to have optimum learning because at every time instant they minimize the weighted error energy from the start of the operation (Tsypkin 1973). These properties are illustrated in the following example.

TABLE 9.7
Summary of RLS and LMS performance in a stationary SOE.

Property	**Growing memory RLS algorithm**	**Exponential memory RLS algorithm**	**LMS algorithm**
Convergence in the mean	For all $n > M$	Asymptotically for $n \to \infty$	Asymptotically for $n \to \infty$
Convergence in MS	Independent of the eigenvalue spread	Independent of the eigenvalue spread	Depends on the eigenvalue spread
Excess MSE	$P_{ex}(n) = \frac{M\sigma_o^2}{n-M-1} \to 0$	$P_{ex}(\infty) = \frac{1-\lambda}{1+\lambda} M\sigma_o^2$	$P_{ex}(\infty) \simeq \mu\sigma_o^2 \text{ tr } \mathbf{R}$

EXAMPLE 9.5.2. Consider the adaptive equalizer of Example 9.4.3 shown in block diagram form in Figure 9.26. In this example, we replace the LMS block in Figure 9.26 by the RLS block, and we study the performance of the RLS algorithm and compare it with that of the LMS algorithm. The input data source is a Bernoulli sequence $\{y(n)\}$ with symbols +1 and −1 having zero mean and unit variance. The channel impulse response is a raised cosine

$$h(n) = \begin{cases} 0.5\left\{1+\cos\left[\frac{2\pi}{W}(n-2)\right]\right\} & n = 1,2,3 \\ 0 & \text{otherwise} \end{cases} \tag{9.5.56}$$

where the parameter W controls the amount of channel distortion [or the eigenvalue spread $\chi(\mathbf{R})$ produced by the channel]. The channel noise sequence $v(n)$ is white Gaussian with $\sigma_v^2 = 0.001$. The adaptive equalizer has $M = 11$ coefficients, and the

input signal $y(n)$ is delayed by $\Delta = 7$ samples. The error signal $e(n) = y(n-\Delta) - \hat{y}(n)$ is used along with $x(n)$ to implement the RLS algorithm given in Table 9.6 with $\mathbf{c}(0) = 0$ and $\delta = 0.001$. We performed Monte Carlo simulations on 100 realizations of random sequences with $W = 2.9$ and $W = 3.5$, and $\lambda = 1$ and 0.8 The results are shown in Figures 9.34 and 9.35.

Effect of eigenvalue spread. Performance plots of the RLS algorithm for $W = 2.9$ and $W = 3.5$ are shown in Figure 9.34. In plot (*a*) we depict MSE learning curves along with the steady-state (or minimum) error. We observe that the MSE convergence rate of the RLS, unlike that for the LMS, does not change with W [or equivalently with change in $\chi(\mathbf{R})$]. The steady-state error, on the other hand, increases with W. The important difference between the two algorithms is that the convergence rate is faster for the RLS (compare Figures 9.34 and 9.27). Clearly, this faster convergence of the RLS algorithm is achieved by an increase in computational complexity. In plots (*b*) and (*c*) we show the ensemble averaged equalizer coefficients. Clearly, the responses are symmetric with respect to $n = 5$, as assumed. Also equalizer coefficients converge to different inverses due to changes in the channel characteristics.

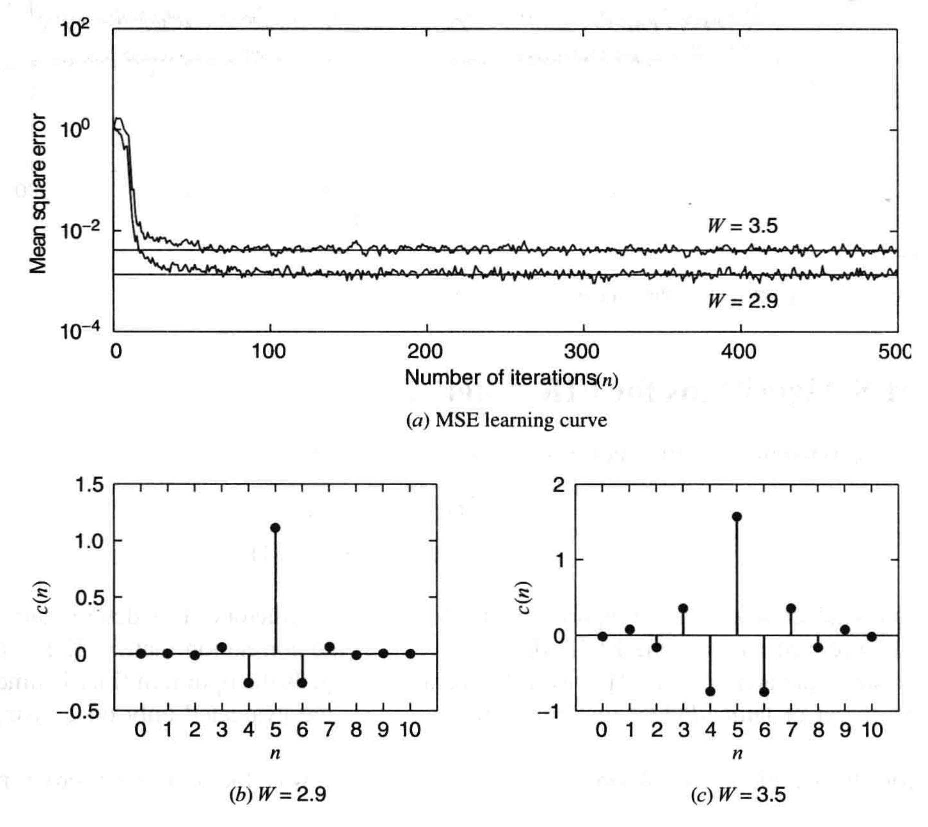


FIGURE 9.34
Performance analysis curves of the RLS algorithm in the adaptive equalizer: $\lambda = 1$.

Effect of forgetting factor λ. In Figure 9.35 we show the MSE learning curves obtained for $W = 2.9$ and with two different factors of 1 and 0.8. For $\lambda = 1$, as explained before, the algorithm has infinite memory and hence the steady-state excess MSE is zero. This fact can be verified in the plot for $\lambda = 1$ in which the MSE converges to the minimum error. For $\lambda = 0.8$, the effective memory is $1/(1-\lambda) = 5$, which clearly is inadequate for the accurate estimation of the required statistics, resulting in increased excess MSE. Therefore, the algorithm should produce a nonzero excess MSE. This fact can be observed from the plot for $\lambda = 0.8$.

There are two practical issues regarding the RLS algorithm that need an explanation. The first issue relates to the practical value of λ. Although λ can take any value in the interval $0 \le \lambda \le 1$, since it influences the effective memory size, the value of λ should be closer to 1. This value is determined by the number of parameters to be estimated and the desired size of the effective memory. Typical values used are between 0.99 and 1 (not 0.8, as we used in this example for demonstration). The second issue deals with the actual computation of matrix $P(n)$. This matrix must be conjugate symmetric and positive definite. However, an implementation of the CRLS algorithm of Table 9.6 on a finite-precision processor will eventually disturb this symmetry and positive definiteness and would result in an unstable performance. Therefore, it is necessary to force this symmetry either by

computing only its lower (or upper) triangular values or by using $P(n) \leftarrow [P(n)+P^{H}(n)]/2$. Failure to do so generally affects the algorithm performance for $\lambda<1$.

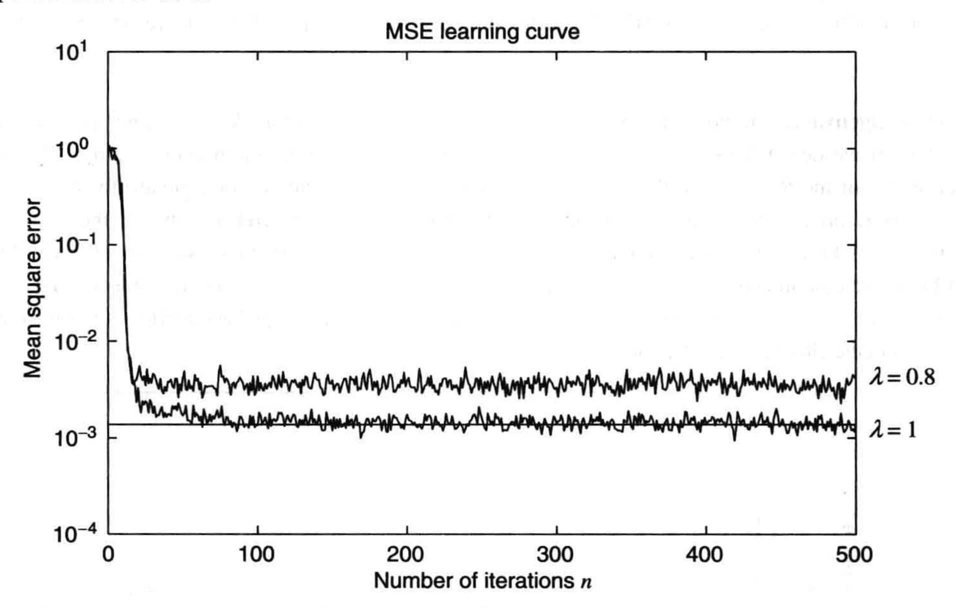


FIGURE 9.35
MSE learning curves of the RLS algorithm in the adaptive equalizer: $W=2.9$.

9.6 Fast RLS Algorithms for FIR Filtering

In Section 6.3 we exploited the shift invariance of the input data vector

$$\boldsymbol{x}_{m+1}(n)=\begin{bmatrix}\boldsymbol{x}_m(n)\\ x(n-m)\end{bmatrix}=\begin{bmatrix}x(n)\\ \boldsymbol{x}_m(n-1)\end{bmatrix} \tag{9.6.1}$$

to develop a lattice-ladder structure for optimum FIR filters and predictors. The determination of the optimum parameters (see Figure 6.3) required the LDL^{H} decomposition of the correlation matrix $\mathbf{R}(n)$ and the solution of three triangular systems at each time n. However, for stationary signals the optimum filter is time-invariant, and the coefficients of its direct or lattice-ladder implementation structure are evaluated only once, using the algorithm of Levinson.

The key for the development of order-recursive algorithms was the following order partitioning of the correlation matrix

$$\boldsymbol{R}_{m+1}(n)=\begin{bmatrix}\boldsymbol{R}_m(n) & \boldsymbol{r}_m^{b}(n)\\ \boldsymbol{r}_m^{bH}(n) & P_x(n-m)\end{bmatrix}=\begin{bmatrix}P_x(n) & \boldsymbol{r}_m^{f\,H}(n)\\ \boldsymbol{r}_m^{f}(n) & \boldsymbol{R}_m(n-1)\end{bmatrix} \tag{9.6.2}$$

which is a result of the shift-invariance property (9.6.1). The same partitioning can be obtained for the LS correlation matrix $\hat{\boldsymbol{R}}_m(n)$

$$\begin{aligned}\hat{\boldsymbol{R}}_{m+1}(n)&=\sum_{j=0}^{n}\lambda^{n-j}\boldsymbol{x}_{m+1}(j)\boldsymbol{x}_{m+1}^{H}(j)\\ &=\begin{bmatrix}\hat{\boldsymbol{R}}_m(n) & \hat{\boldsymbol{r}}_m^{b}(n)\\ \hat{\boldsymbol{r}}_m^{bH}(n) & E_x(n-m)\end{bmatrix}=\begin{bmatrix}E_x(n) & \hat{\boldsymbol{r}}_m^{f\,H}(n)\\ \hat{\boldsymbol{r}}_m^{f}(n) & \hat{\boldsymbol{R}}_m(n-1)\end{bmatrix}\end{aligned} \tag{9.6.3}$$

if we assume that $\boldsymbol{x}_m(-1)=0$, a condition known as *prewindowing* (see Section 7.3). This condition is neccesary to ensure the presence of the term $\hat{\boldsymbol{R}}_m(n-1)$ in the lower right corner partitioning of $\hat{\boldsymbol{R}}_{m+1}(n)$.

The identical forms of (9.6.2) and (9.6.3) imply that the order-recursive relations and the lattice-ladder structure developed in Section 6.3 for optimum FIR filters can be used for *prewindowed* LS FIR filters. Simply, the expectation operator $E\{(\cdot)\}$ should be replaced by the time-averaging operator $\sum_{j=0}^{n} \lambda^{n-j}(\cdot)$, and the term *power* should be replaced by the term *energy*, when we go from the optimum MSE to the LSE formulation.

In this section we exploit the shift invariance (9.6.1) and the time updating

$$\hat{\boldsymbol{R}}_m(n) = \lambda \hat{\boldsymbol{R}}_m(n-1) + \boldsymbol{x}_m(n)\boldsymbol{x}_m^{\mathrm{H}}(n) \tag{9.6.4}$$

to develop the following types of fast algorithms with $O(M)$ complexity:

1. Fast *fixed-order* algorithms for RLS direct-form FIR filters by explicitly updating the gain vectors $\boldsymbol{g}(n)$ and $\bar{\boldsymbol{g}}(n)$.
2. Fast *order-recursive* algorithms for RLS FIR lattice-ladder filters by indirect or direct updating of their coefficients.
3. QR decomposition–based RLS lattice-ladder algorithms using the Givens rotation.

All relationships in Section 6.3 are valid for the prewindowed LS problem, but we replace *P* by *E* to emphasize the energy interpretation of the cost function. The quantities appearing in the partitionings given by (9.6.3) specify a prewindowed LS forward linear predictor $-\boldsymbol{a}_m$ and an LS backward linear predictor $-\boldsymbol{b}_m$. Table 9.8 shows the correspondences between general FIR filtering, FLP, and BLP. Using these correspondences and the normal equations for LS filtering, we can easily obtain the normal equations and the total LSE for the FLP and the BLP, which are also summarized in Table 9.8 (see Problem 9.28). We stress that the predictor parameters $\boldsymbol{a}_m(n)$ and $\boldsymbol{b}_m(n)$ are held fixed over the optimization interval $0 \le j \le n$.

TABLE 9.8
Summary and correspondences between LS FIR filtering, forward linear prediction, and backward linear prediction.

	FIR filter	FLP	BLP
Input data vector	$\boldsymbol{x}_m(n)$	$\boldsymbol{x}_m(n-1)$	$\boldsymbol{x}_m(n)$
Desired response	$y(n)$	$x(n)$	$x(n-m)$
Coefficient vector	$\boldsymbol{c}_m(n)$	$-\boldsymbol{a}_m(n)$	$-\boldsymbol{b}_m(n)$
Error	$\varepsilon_m(n) = y(n) - \boldsymbol{c}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n)$	$\varepsilon_m^f(n) = x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n-1)$	$\varepsilon_m^b(n) = x(n-m) + \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n)$
Cost function	$E_m(n) = \sum_{j=0}^{n} \lambda^{n-j} \lvert \varepsilon_m(j) \rvert^2$	$E_m^f(n) = \sum_{j=0}^{n} \lambda^{n-j} \lvert \varepsilon_m^f(j) \rvert^2$	$E_m^b(n) = \sum_{j=0}^{n} \lambda^{n-j} \lvert \varepsilon_m^b(j) \rvert^2$
Normal equations	$\hat{\boldsymbol{R}}_m(n)\boldsymbol{c}_m(n) = \hat{\boldsymbol{d}}_m(n)$	$\hat{\boldsymbol{R}}_m(n-1)\boldsymbol{a}_m(n) = -\hat{\boldsymbol{r}}_m^f(n)$	$\hat{\boldsymbol{R}}_m(n)\boldsymbol{b}_m(n) = -\hat{\boldsymbol{r}}_m^b(n)$
LSE	$E_m(n) = E_y(n) - \boldsymbol{c}_m^{\mathrm{H}}(n)\hat{\boldsymbol{d}}_m(n)$	$E_m^f(n) = E_x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n)\hat{\boldsymbol{r}}_m^f(n)$	$E_m^b(n) = E_x(n-m) + \boldsymbol{b}_m^{\mathrm{H}}(n)\hat{\boldsymbol{r}}_m^b(n)$
Correlation matrix	$\hat{\boldsymbol{R}}_m(n) = \sum_{j=0}^{n} \lambda^{n-j}\boldsymbol{x}_m(j)\boldsymbol{x}_m^{\mathrm{H}}(j)$	$\hat{\boldsymbol{R}}_m(n-1)$	$\hat{\boldsymbol{R}}_m(n)$
Cross-correlation vectors	$\hat{\boldsymbol{d}}_m(n) = \sum_{j=0}^{n} \lambda^{n-j}\boldsymbol{x}_m(j)y^*(j)$	$\hat{\boldsymbol{r}}_m^f(n) = \sum_{j=0}^{n} \lambda^{n-j}\boldsymbol{x}_m(j-1)x^*(j)$	$\hat{\boldsymbol{r}}_m^b(n) = \sum_{j=0}^{n} \lambda^{n-j}\boldsymbol{x}_m(j)x^*(j-m)$

Table 9.9 summarizes the a priori and a posteriori time updates for the LS FIR filter derived in Section 9.5. If we use the correspondences between general FIR filtering and linear prediction, we can easily deduce similar time-updating recursions for the FLP and the BLP. These updates, which are also discussed in Problem 9.29, are summarized in Table 9.9.

TABLE 9.9
Summary of LS time-updating relations using a priori and a posteriori errors.

	Equation	A priori time updating	A posteriori time updating
Gain	(a)	$\hat{\boldsymbol{R}}_m(n)\boldsymbol{g}_m(n) = \boldsymbol{x}_m(n)$	$\lambda\hat{\boldsymbol{R}}_m(n-1)\bar{\boldsymbol{g}}_m(n) = \boldsymbol{x}_m(n)$
Filter	(b)	$e_m(n) = y(n) - \boldsymbol{c}_m^{\mathrm{H}}(n-1)\boldsymbol{x}_m(n)$	$\varepsilon_m(n) = y(n) - \boldsymbol{c}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n)$
	(c)	$\boldsymbol{c}_m(n) = \boldsymbol{c}_m(n-1) + \boldsymbol{g}_m(n)e_m^*(n)$	$\boldsymbol{c}_m(n) = \boldsymbol{c}_m(n-1) + \bar{\boldsymbol{g}}_m(n)\varepsilon_m^*(n)$
	(d)	$E_m(n) = \lambda E_m(n-1) + \alpha_m(n)\lvert e_m(n)\rvert^2$	$E_m(n) = \lambda E_m(n-1) + \dfrac{\lvert\varepsilon_m(n)\rvert^2}{\alpha_m(n)}$
FLP	(e)	$e_m^f(n) = x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n-1)\boldsymbol{x}_m(n-1)$	$\varepsilon_m^f(n) = x(n) + \boldsymbol{a}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n-1)$
	(f)	$\boldsymbol{a}_m(n) = \boldsymbol{a}_m(n-1) - \boldsymbol{g}_m(n-1)e_m^{f*}(n)$	$\boldsymbol{a}_m(n) = \boldsymbol{a}_m(n-1) - \bar{\boldsymbol{g}}_m(n-1)\varepsilon_m^{f*}(n)$
	(g)	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1)\lvert e_m^f(n)\rvert^2$	$E_m^{\mathrm{f}}(n) = \lambda E_m^{\mathrm{f}}(n-1) + \dfrac{\lvert\varepsilon_m^{\mathrm{f}}(n)\rvert^2}{\alpha_m(n-1)}$
BLP	(h)	$e_m^b(n) = x(n-m) + \boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{x}_m(n)$	$\varepsilon_m^b(n) = x(n-m) + \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n)$
	(i)	$\boldsymbol{b}_m(n) = \boldsymbol{b}_m(n-1) - \boldsymbol{g}_m(n)e_m^{b*}(n)$	$\boldsymbol{b}_m(n) = \boldsymbol{b}_m(n-1) - \bar{\boldsymbol{g}}_m(n)\varepsilon_m^{b*}(n)$
	(j)	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n)\lvert e_m^b(n)\rvert^2$	$E_m^{\mathrm{b}}(n) = \lambda E_m^{\mathrm{b}}(n-1) + \dfrac{\lvert\varepsilon_m^{\mathrm{b}}(n)\rvert^2}{\alpha_m(n)}$

9.6.1 Fast Fixed-Order RLS FIR Filters

The major computational task in RLS filters is the computation of the gain vector $\boldsymbol{g}(n)$ or $\bar{\boldsymbol{g}}(n)$. The CRLS algorithm updates the inverse matrix $\hat{\boldsymbol{R}}^{-1}(n)$ and then determines the gain vector via a matrix-by-vector multiplication that results in $O(M^2)$ complexity. The only way to reduce the complexity from $O(M^2)$ to $O(M)$ is by directly updating the gain vectors. We next show how to develop such algorithms by exploiting the shift-invariant structure of the input data vector shown in (9.6.1).

Fast Kalman algorithm: Updating the gain $\boldsymbol{g}(n)$
Suppose that we know the gain

$$\boldsymbol{g}_m(n-1) = \hat{\boldsymbol{R}}_m^{-1}(n-1)\boldsymbol{x}_m(n-1) \tag{9.6.5}$$

and we wish to compute the gain

$$\boldsymbol{g}_m(n) = \hat{\boldsymbol{R}}_m^{-1}(n)\boldsymbol{x}_m(n) \tag{9.6.6}$$

at the next time instant by "adjusting" $\boldsymbol{g}_m(n-1)$, using the new data $\{\boldsymbol{x}_m(n), y(n)\}$.

If we use the matrix inversion by partitioning formulas (6.1.24) and (6.1.26) for matrix $\hat{\boldsymbol{R}}_{m+1}(n)$, we have

$$\hat{\boldsymbol{R}}_{m+1}(n) = \begin{bmatrix} \hat{\boldsymbol{R}}_m^{-1}(n) & \boldsymbol{0}_m \\ \boldsymbol{0}_m^H & 0 \end{bmatrix} + \frac{1}{E_m^{\mathrm{b}}(n)} \begin{bmatrix} \boldsymbol{b}_m(n) \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_m^{\mathrm{H}}(n) & 1 \end{bmatrix} \tag{9.6.7}$$

and

$$\hat{R}_{m+1}(n) = \begin{bmatrix} 0 & \boldsymbol{0}_m^{\mathrm{H}} \\ \boldsymbol{0}_m & \hat{\boldsymbol{R}}_m^{-1}(n) \end{bmatrix} + \frac{1}{E_m^{\mathrm{f}}(n)} \begin{bmatrix} 1 \\ \boldsymbol{a}_m(n) \end{bmatrix} \begin{bmatrix} 1 & \boldsymbol{a}_m^{\mathrm{H}}(n) \end{bmatrix} \tag{9.6.8}$$

as was shown in section 6.1.

Using (9.6.7), the first partitioning in (9.6.1), and the definition of $\varepsilon_m^b(n)$ from Table 9.9, we obtain

$$\boldsymbol{g}_{m+1}(n) = \begin{bmatrix} \boldsymbol{g}_m(n) \\ 0 \end{bmatrix} + \frac{\varepsilon_m^{\mathrm{b}}(n)}{E_m^{\mathrm{b}}(n)} \begin{bmatrix} \boldsymbol{b}_m(n) \\ 1 \end{bmatrix} \tag{9.6.9}$$

which provides a *pure order update* of the gain vector $\boldsymbol{g}_m(n)$. Similarly, using (9.6.8), the second partitioning in (9.6.1), and the definition of $\varepsilon_m^f(n)$ from Table 9.9, we have

$$\boldsymbol{g}_{m+1}(n) = \begin{bmatrix} 0 \\ \boldsymbol{g}_m(n-1) \end{bmatrix} + \frac{\varepsilon_m^{\mathrm{f}}(n)}{E_m^{\mathrm{f}}(n)} \begin{bmatrix} 1 \\ \boldsymbol{a}_m(n) \end{bmatrix} \tag{9.6.10}$$

which provides a *combined order and time update* of the gain vector $\boldsymbol{g}_m(n)$. This is the key to the development of fast algorithms for updating the gain vector.

Given the gain $\boldsymbol{g}_m(n-1)$, first we compute $\boldsymbol{g}_{m+1}(n)$, using (9.6.10). Then we compute $\boldsymbol{g}_m(n)$ from the first m equations of (9.6.9) as

$$\boldsymbol{g}_m(n) = \boldsymbol{g}_{m+1}^{\lceil m \rceil}(n) - g_{m+1}^{(m+1)}(n)\boldsymbol{b}_m(n) \tag{9.6.11}$$

because

$$g_{m+1}^{(m+1)}(n) = \frac{\varepsilon_m^{\mathrm{b}}(n)}{E_m^{\mathrm{b}}(n)} \tag{9.6.12}$$

from the last equation in (9.6.9). The updatings (9.6.9) and (9.6.10) require time updatings for the predictors $\boldsymbol{a}_m(n)$ and $\boldsymbol{b}_m(n)$ and the minimum error energies $E_m^f(n)$ and $E_m^b(n)$, which are given in Table 9.9. The only remaining problem is the coupling between $\boldsymbol{g}_m(n)$ in (9.6.11) and $\boldsymbol{b}_m(n)$ in

$$\boldsymbol{b}_m(n) = \boldsymbol{b}_m(n-1) - \boldsymbol{g}_m(n)e_m^{\mathrm{b}*}(n) \tag{9.6.13}$$

which can be avoided by eliminating $\boldsymbol{b}_m(n)$. Carrying out the elimination, we obtain

$$\boldsymbol{g}_m(n) = \frac{\boldsymbol{g}_{m+1}^{\lceil m \rceil}(n) - g_{m+1}^{(m+1)}(n)\boldsymbol{b}_m(n-1)}{1 - g_{m+1}^{(m+1)}(n)e_m^{\mathrm{b}*}(n)} \tag{9.6.14}$$

which provides the last step required to complete the updating. This approach, which is known as the *fast Kalman algorithm*, was developed in Falconer and Ljung (1978) using the ideas introduced by Morf (1974). To emphasize the fixed-order nature of the algorithm, we set $m = M$ and drop the order subscript for all quantities of order M. The computational organization of the algorithm, which requires 9M operations per time updating, is summarized in Table 9.10.

TABLE 9.10
Fast Kalman algorithm for time updating of LS FIR filters.

Equation	Computation
	Old estimates: $\boldsymbol{a}(n-1), \boldsymbol{b}(n-1), \boldsymbol{g}(n-1), \boldsymbol{c}(n-1), E^f(n-1)$
	New data: $\{\boldsymbol{x}(n), y(n)\}$
	Gain and predictor update
(*a*)	$e^f(n) = x(n) + \boldsymbol{a}^{\mathrm{H}}(n-1)\boldsymbol{x}(n-1)$
(*b*)	$\boldsymbol{a}(n) = \boldsymbol{a}(n-1) - \boldsymbol{g}(n-1)e^{f*}(n)$
(*c*)	$\varepsilon^f(n) = x(n) + \boldsymbol{a}^{\mathrm{H}}(n)\boldsymbol{x}(n-1)$
(*d*)	$E^f(n) = \lambda E^f(n-1) + \varepsilon^f(n)e^{f*}(n)$
(*e*)	$\boldsymbol{g}_{M+1}(n) = \begin{bmatrix} 0 \\ \boldsymbol{g}(n-1) \end{bmatrix} + \frac{\varepsilon^f(n)}{E^f(n)} \begin{bmatrix} 1 \\ \boldsymbol{a}(n) \end{bmatrix}$
(*f*)	$e^b(n) = x(n-M) + \boldsymbol{b}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)$
(*g*)	$\boldsymbol{g}(n) = \frac{\boldsymbol{g}_{M+1}^{\lceil M \rceil}(n) - g_{M+1}^{(M+1)}(n)\boldsymbol{b}(n-1)}{1 - g_{M+1}^{(M+1)}(n)e^{b*}(n)}$
(*h*)	$\boldsymbol{b}(n) = \boldsymbol{b}(n-1) - \boldsymbol{g}(n)e^{b*}(n)$
	Filter update
(*i*)	$e(n) = y(n) - \boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)$
(*j*)	$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \boldsymbol{g}(n)e^*(n)$

The FAEST algorithm: Updating the gain $\bar{g}(n)$

In a similar way we can update the gain vector

$$\bar{g}_m(n) = \frac{1}{\lambda}\hat{R}_m^{-1}(n-1)x_m(n) \tag{9.6.15}$$

by using (9.6.9) and (9.6.10). Indeed, using (9.6.10) with the lower partitioning (9.6.1) and (9.6.9) with the upper partitioning (9.6.1), we obtain

$$\bar{g}_{m+1}(n) = \begin{bmatrix} 0 \\ \bar{g}_m(n-1) \end{bmatrix} + \frac{e_m^{\mathrm{f}}(n)}{\lambda E_m^{\mathrm{f}}(n-1)} \begin{bmatrix} 1 \\ a_m(n-1) \end{bmatrix} \tag{9.6.16}$$

and

$$\bar{g}_{m+1}(n) = \begin{bmatrix} \bar{g}_m(n) \\ 0 \end{bmatrix} + \frac{e_m^{\mathrm{b}}(n)}{\lambda E_m^{\mathrm{b}}(n-1)} \begin{bmatrix} b_m(n-1) \\ 1 \end{bmatrix} \tag{9.6.17}$$

which provide a link between $\bar{g}_m(n-1)$ and $\bar{g}_m(n)$. From (9.6.17) we obtain

$$\bar{g}_m(n) = \bar{g}_{m+1}^{[m]}(n) - \bar{g}_{m+1}^{(m+1)}(n)b_m(n-1) \tag{9.6.18}$$

because

$$\bar{g}_{m+1}^{(m+1)}(n) = \frac{e_m^{\mathrm{b}}(n)}{\lambda E_m^{\mathrm{b}}(n-1)} \tag{9.6.19}$$

from the last row of (9.6.17). The fundamental difference between (9.6.9) and (9.6.17) is that the presence of $b_m(n-1)$ in the latter breaks the coupling between gain vector and backward predictor. Furthermore, (9.6.19) can be used to compute $e_m^b(n)$ by

$$e_m^b(n) = \lambda E_m^b(n-1)\bar{g}_{m+1}^{(m+1)}(n) \tag{9.6.20}$$

with only two multiplications.

The time updatings of the predictors using the gain $\bar{g}_m(n)$, which are given in Table 9.9, require the a posteriori errors that can be computed from the a priori errors by using the conversion factor

$$\bar{\alpha}_m(n) = 1 + \bar{g}_m^{\mathrm{H}}(n)x_m(n) \tag{9.6.21}$$

which should be updated in time as well. This can be achieved by a two-step procedure as follows. First, using (9.6.16) and the lower partitioning (9.6.1), we obtain

$$\bar{\alpha}_{m+1}(n) = \bar{\alpha}_m(n-1) + \frac{|e_m^f(n)|^2}{\lambda E_m^f(n-1)} \tag{9.6.22}$$

which is a combined time and order updating. Then we use (9.6.17) and the upper partitioning (9.6.1) to obtain

$$\bar{\alpha}_m(n) = \bar{\alpha}_{m+1}(n) - \bar{g}_{m+1}^{(m+1)}(n)e_m^{\mathrm{b}*}(n) \tag{9.6.23}$$

or

$$\bar{\alpha}_m(n) = \bar{\alpha}_{m+1}(n) - \frac{|e_m^{\mathrm{b}}(n)|^2}{\lambda E_m^{\mathrm{b}}(n-1)} \tag{9.6.24}$$

which in conjunction with (9.6.22) provides the required time update $\bar{\alpha}_m(n-1) \to \bar{\alpha}_{m+1}(n) \to \bar{\alpha}_m(n)$.

This leads to the *fast a posteriori error sequential technique (FAEST)* algorithm presented in Table 9.11, which was introduced in Carayannis et al. (1983). The FAEST algorithm requires only $7M$ operations per time update and is the most efficient known algorithm for prewindowed RLS FIR filters.

Fast transversal filter (FTF) algorithm. This is an a posteriori type of algorithm obtained from the FAEST by

using the conversion factor

$$\alpha_m(n) = 1 - \boldsymbol{g}_m^{\mathrm{H}}(n)\boldsymbol{x}_m(n) \tag{9.6.25}$$

instead of the conversion factor $\bar{\alpha}_m(n) = 1/\alpha_m(n)$. Using the Levinson recursions (9.6.9) and (9.6.10) in conjunction with the upper and lower partitionings in (9.6.10), we obtain

$$\alpha_{m+1}(n) = \alpha_m(n) - \frac{|\varepsilon_m^{\mathrm{b}}(n)|^2}{E_m^{\mathrm{b}}(n)} \tag{9.6.26}$$

and

$$\alpha_{m+1}(n) = \alpha_m(n-1) - \frac{|\varepsilon_m^{\mathrm{f}}(n)|^2}{E_m^{\mathrm{f}}(n)} \tag{9.6.27}$$

respectively. To obtain the FTF algorithm, we replace $\bar{\alpha}_m(n)$ in Table 9.11 by $1/\alpha_m(n)$ and Equation (*h*) by (9.6.27). To obtain $\alpha_m(n)$ from $\alpha_{m+1}(n)$, we cannot use (9.6.26) because it requires quantities dependent on $\alpha_m(n)$. To avoid this problem, we replace Equation (*i*) by the following relation

$$\alpha_m(n) = \frac{\alpha_{m+1}(n)}{1 - \alpha_{m+1}(n)\bar{g}_{m+1}^{(m+1)}(n)e_m^{\mathrm{b}*}(n)} \tag{9.6.28}$$

obtained by combining (9.6.24), (9.6.19), and $\bar{\alpha}_m(n) = 1/\alpha_m(n)$. This algorithm, which has the same complexity as FAEST, was introduced in Cioffi and Kailath (1984) using a geometric derivation, and is known as the *fast transversal filter (FTF) algorithm.*

TABLE 9.11
FAEST algorithm for time updating of LS FIR filters.

Equation	Computation
	Old estimates: $\boldsymbol{a}(n-1), \boldsymbol{b}(n-1), \boldsymbol{c}(n-1), \bar{\boldsymbol{g}}(n-1),\ E^f(n-1), E^b(n-1), \bar{\alpha}(n-1)$ New data: $\{\boldsymbol{x}(n), y(n)\}$
	Gain and predictor update
(*a*)	$e^f(n) = x(n) + \boldsymbol{a}^{\mathrm{H}}(n-1)\boldsymbol{x}(n-1)$
(*b*)	$\varepsilon^f(n) = \dfrac{e^f(n)}{\bar{\alpha}(n-1)}$
(*c*)	$\boldsymbol{a}(n) = \boldsymbol{a}(n-1) - \bar{\boldsymbol{g}}(n-1)\varepsilon^{f*}(n)$
(*d*)	$E^f(n) = \lambda E^f(n-1) + \varepsilon^f(n)e^{f*}(n)$
(*e*)	$\bar{\boldsymbol{g}}_{M+1}(n) = \begin{bmatrix} 0 \\ \bar{\boldsymbol{g}}(n-1) \end{bmatrix} + \dfrac{e^f(n)}{\lambda E^f(n-1)} \begin{bmatrix} 1 \\ \boldsymbol{a}(n-1) \end{bmatrix}$
(*f*)	$e^b(n) = \lambda E^b(n-1)\bar{g}_{M+1}^{(M+1)}(n)$
(*g*)	$\bar{\boldsymbol{g}}(n) = \bar{\boldsymbol{g}}_{M+1}^{[M]}(n) - \bar{g}_{M+1}^{(M+1)}(n)\boldsymbol{b}(n-1)$
(*h*)	$\bar{\alpha}_{M+1}(n) = \bar{\alpha}(n-1) + \dfrac{\lvert e^{\mathrm{f}}(n)\rvert^2}{\lambda E^{\mathrm{f}}(n-1)}$
(*i*)	$\bar{\alpha}(n) = \bar{\alpha}_{M+1}(n) - \bar{g}_{M+1}^{(M+1)*}(n)e^{\mathrm{b}}(n)$
(*j*)	$\boldsymbol{b}(n) = \boldsymbol{b}(n-1) - \bar{\boldsymbol{g}}(n)\varepsilon^{\mathrm{b}*}(n)$
(*k*)	$\varepsilon^{\mathrm{b}}(n) = \dfrac{e^{\mathrm{b}}(n)}{\bar{\alpha}(n)}$
(*l*)	$E^{\mathrm{b}}(n) = \lambda E^{\mathrm{b}}(n-1) + \varepsilon^{\mathrm{b}}(n)e^{\mathrm{b}*}(n)$
	Filter update
(*m*)	$e(n) = y(n) - \boldsymbol{c}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)$
(*n*)	$\varepsilon(n) = \dfrac{e(n)}{\bar{\alpha}(n)}$
(*o*)	$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \bar{\boldsymbol{g}}(n)\varepsilon^*(n)$

An alternative updating to (9.6.27) can be obtained by noticing that

$$\alpha_{m+1}(n) = \alpha_m(n-1) - \alpha_m^2(n-1)\frac{|e_m^{\mathrm{f}}(n)|^2}{E_m^{\mathrm{f}}(n)}$$

$$= \frac{\alpha_m(n-1)}{E_m^{\mathrm{f}}(n)}[E_m^{\mathrm{f}}(n) - \alpha_m(n-1)|e_m^{\mathrm{f}}(n)|^2]$$

or equivalently

$$\alpha_{m+1}(n) = \alpha_m(n-1)\frac{\lambda E_m^{\mathrm{f}}(n-1)}{E_m^{\mathrm{f}}(n)} \tag{9.6.29}$$

which can be used instead of (9.6.27) in the FTF algorithm. In a similar way, we can show that

$$\alpha_{m+1}(n) = \alpha_m(n)\frac{\lambda E_m^{\mathrm{b}}(n-1)}{E_m^{\mathrm{b}}(n)} \tag{9.6.30}$$

which will be used later.

Some practical considerations

Figure 9.36 shows the realization of an adaptive RLS filter using the direct-form structure. The coefficient updating can be done using any of the introduced fast RLS algorithms. Some issues related to the implementation of these filters using multiprocessing are discussed in Problem 9.48.

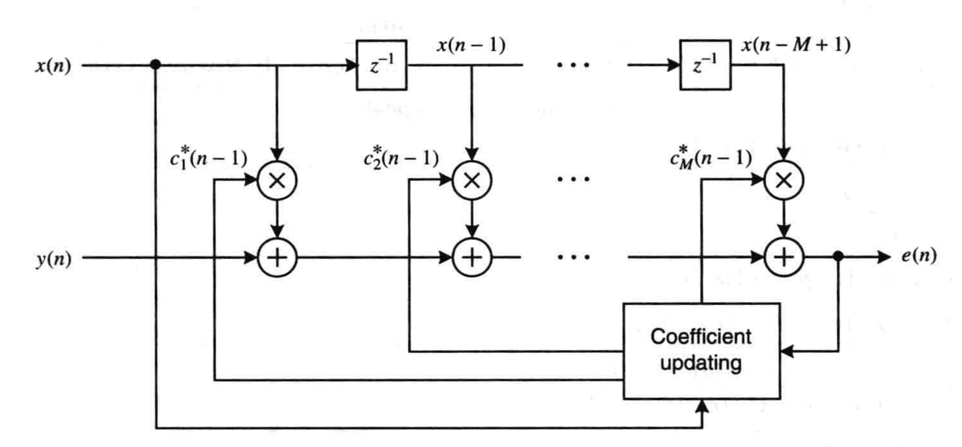


FIGURE 9.36
Implementation of an adaptive FIR filter using a direct-form structure.

In practice, the fast direct-form RLS algorithms are initialized at $n=0$ by setting

$$E^{\mathrm{f}}(-1) = E^{\mathrm{b}}(-1) = \delta > 0$$
$$\alpha(-1) = 1 \quad \text{or} \quad \bar{\alpha}(-1) = 1 \tag{9.6.31}$$

and all other quantities equal to zero. The constant δ is chosen as a small positive number on the order of $0.01\sigma_x^2$ (Hubing and Alexander 1991). For $\lambda < 1$, the effects of the initial conditions are quickly "forgotten."

Although the fast direct-form RLS algorithms have the lowest computational complexity, they suffer from numerical instability when $\lambda < 1$ (Ljung and Ljung 1985). When these algorithms are implemented with finite precision, the exact algebraic relations used for their derivation breakdown and lead to numerical problems.

There are two ways to deal with stabilization of the fast direct-form RLS algorithms. In the first approach, we try to identify precursors of ill behavior (warnings) and then use appropriate rescue operations to restore the normal operation of the algorithm (Lin 1984; Cioffi and Kailath 1984). One widely used rescue variable is

$$\eta_m(n) \triangleq \frac{\alpha_{m+1}(n)}{\alpha_m(n)} = \frac{\lambda E_m^{\mathrm{b}}(n-1)}{E_m^{\mathrm{b}}(n)} \tag{9.6.32}$$

which satisfies $0 \le \eta_m(n) \le 1$ for infinite-precision arithmetic.

In the second approach, we exploit the fact that certain algorithmic quantities can be computed in two different ways. Therefore, we could use their difference, which provides a measure of the numerical errors, to change the dynamics of the error propagation system and stabilize the algorithm. For example, both $e_m^b(n)$ and $\alpha_m(n)$ can be computed either using their definition or simpler order-recursions. This approach has been used to obtain stabilized algorithms with complexities 9*M* and 8*M*; however, their performance is highly dependent on proper initialization (Slock and Kailath 1991, 1993).

9.6.2 RLS Lattice-Ladder Filters

The lattice-ladder structure[1] derived in Section 6.3 using the MSE criterion, due to the similarity of (9.6.2) and (9.6.3), holds for the prewindowed LSE criterion as well. This structure, which is depicted in Figure 9.37 for the a posteriori error case, is described by the following equations

$$\varepsilon_0^{\mathrm{f}}(n) = \varepsilon_0^{\mathrm{b}}(n) = x(n)$$

$$\varepsilon_{m+1}^{\mathrm{f}}(n) = \varepsilon_m^{\mathrm{f}}(n) + k_m^{\mathrm{f}*}(n)\varepsilon_m^{\mathrm{b}}(n-1) \qquad 0 \le m < M-1 \tag{9.6.33}$$

$$\varepsilon_{m+1}^{\mathrm{b}}(n) = \varepsilon_m^{\mathrm{b}}(n-1) + k_m^{\mathrm{b}*}(n)\varepsilon_m^{\mathrm{f}}(n) \qquad 0 \le m < M-1 \tag{9.6.34}$$

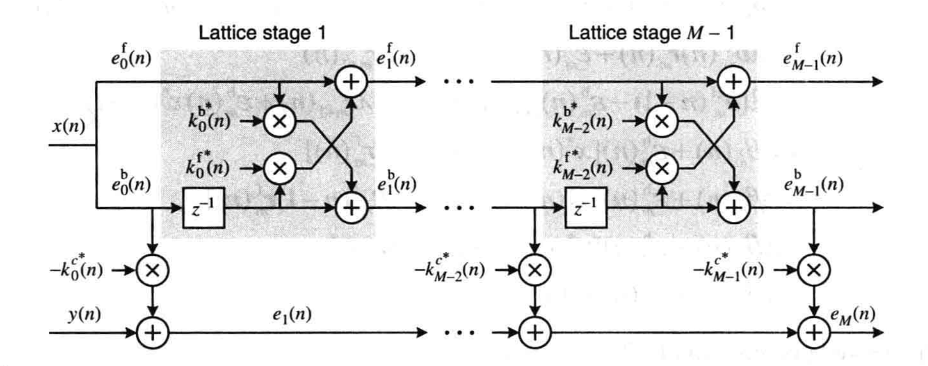


FIGURE 9.37
A posteriori error RLS lattice-ladder filter.

for the lattice part and

$$\begin{aligned} \varepsilon_0(n) &= y(n) \\ \varepsilon_{m+1}(n) &= \varepsilon_m(n) - k_m^{c*}(n)\varepsilon_m^{\mathrm{b}}(n) \qquad 0 \le m \le M-1 \end{aligned} \tag{9.6.35}$$

for the ladder part. The lattice parameters are given by

$$k_m^{\mathrm{f}}(n) = -\frac{\beta_m(n)}{E_m^{\mathrm{b}}(n-1)} \tag{9.6.36}$$

and

$$k_m^{\mathrm{b}}(n) = -\frac{\beta_m^*(n)}{E_m^{\mathrm{f}}(n)} \tag{9.6.37}$$

and the ladder parameters by

$$k_m^c(n) = \frac{\beta_m^c(n)}{E_m^{\mathrm{b}}(n)} \tag{9.6.38}$$

[1]In Chapter 6 we used the symbol $e(n)$ because we had no need to distinguish between a priori and a posteriori errors. However, since the error $e(n)$ in Section 6.3 is an a posteriori error, we now use the symbol $\varepsilon(n)$.

where
$$\beta_m(n) = \boldsymbol{b}_m^{\mathrm{H}}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n) + r_{m+1}^{\mathrm{f}}(n) \tag{9.6.39}$$

and
$$\beta_m^c(n) = \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{d}_m(n) + d_{m+1}(n) \tag{9.6.40}$$

are the partial correlation parameters.

However, as we recall, the time updating of the minimum LSE energies and the partial correlations is possible only if there is a time update for the correlation matrix $\boldsymbol{R}_m(n)$ and the cross-correlation vector $\boldsymbol{d}_m(n)$.

The minimum LSE energies can be updated in time using

$$E_m^f(n) = \lambda E_m^f(n-1) + e_m^f(n)\varepsilon_m^{f*}(n) \tag{9.6.41}$$

$$E_m^b(n) = \lambda E_m^b(n-1) + e_m^b(n)\varepsilon_m^{b*}(n) \tag{9.6.42}$$

or their variations, given in Table 9.9.

To update the partial correlation $\beta_m(n)$, we start with the definition (9.6.39) and then use the time-updating formulas for all involved quantities, rearranging and recombining terms as follows:

$$\begin{aligned}
\beta_m(n+1) &= \boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{r}_m^{\mathrm{f}}(n+1) + r_{m+1}^{\mathrm{f}}(n+1) \\
&= \boldsymbol{b}_m^{\mathrm{H}}(n)[\lambda\boldsymbol{r}_m^{\mathrm{f}}(n) + \boldsymbol{x}_m(n)x^*(n+1)] + [\lambda r_{m+1}^{\mathrm{f}}(n) + x(n-m)x^*(n+1)] \\
&= \lambda\boldsymbol{b}_m^{\mathrm{H}}(n)\boldsymbol{r}_m^{\mathrm{f}}(n) + \varepsilon_m^{\mathrm{b}}(n)x^*(n+1) + \lambda r_{m+1}^{\mathrm{f}}(n) \\
&= \lambda[\boldsymbol{b}_m^{\mathrm{H}}(n-1) - \varepsilon_m^{\mathrm{b}}(n)\overline{\boldsymbol{g}}_m(n)]\boldsymbol{r}_m^{\mathrm{f}}(n) + \lambda r_{m+1}^{\mathrm{f}}(n) + \varepsilon_m^{\mathrm{b}}(n)x^*(n+1) \\
&= \lambda\beta_m(n) + \varepsilon_m^{\mathrm{b}}(n)[x^*(n+1) - \lambda\overline{\boldsymbol{g}}_m(n)\boldsymbol{r}_m^{\mathrm{f}}(n)] \\
&= \lambda\beta_m(n) + \varepsilon_m^{\mathrm{b}}(n)[x^*(n+1) - \boldsymbol{x}_m^{\mathrm{H}}(n)\boldsymbol{R}_m^{-1}(n-1)\boldsymbol{r}_m^{\mathrm{f}}(n)] \\
&= \lambda\beta_m(n) + \varepsilon_m^{\mathrm{b}}(n)[x^*(n+1) + \boldsymbol{x}_m^{\mathrm{H}}(n)\boldsymbol{a}(n)] \\
&= \lambda\beta_m(n) + \varepsilon_m^{\mathrm{b}}(n)e_m^{\mathrm{f}*}(n+1)
\end{aligned}$$

which provides the desired update formula. The updating

$$\beta_m(n) = \lambda\beta_m(n-1) + \varepsilon_m^{\mathrm{b}}(n-1)e_m^{\mathrm{f}*}(n) \tag{9.6.43}$$

$$= \lambda\beta_m(n-1) + \frac{1}{\alpha_m(n-1)}\varepsilon_m^{\mathrm{b}}(n-1)\varepsilon_m^{\mathrm{f}*}(n) \tag{9.6.44}$$

is feasible because the right-hand side involves already-known quantities.

In a similar way (see Problem 9.36), we can show that

$$\beta_m^c(n) = \lambda\beta_m^c(n-1) + \varepsilon_m^{\mathrm{b}}(n)e_m^*(n) \tag{9.6.45}$$

$$= \lambda\beta_m^c(n-1) + \frac{1}{\alpha_m(n)}\varepsilon_m^{\mathrm{b}}(n)\varepsilon_m^*(n) \tag{9.6.46}$$

which facilitates the updating of the ladder parameters.

To obtain an a posteriori algorithm, we need the conversion factor $\alpha_m(n)$, which can be obtained using the order-recursive formula (9.6.26). A detailed organization of the a posteriori LS lattice-ladder algorithm, which requires about 20*M* operations per time update, is given in Table 9.12. The initialization of the algorithm is easily obtained from the definitions of the corresponding quantities. The condition $\alpha_0(n-1)=1$ follows from (9.6.25), and the positive constant δ is chosen to ensure the inveribility of the LS correlation matrix $\boldsymbol{R}(n)$ (see Section 9.5).The time-updating recursions (*c*) and (*d*) can be replaced by order recursions, as explained in Problem 9.37.

TABLE 9.12
Computational organization of a posteriori LS lattice-ladder algorithm.

Equation	Computation
	Time initialization ($n = 0$)
	$E_m^f(-1) = E_m^b(-1) = \delta > 0 \quad 0 \le m < M-1$
	$\beta_m(-1) = 0, \varepsilon_m^b(-1) = 0 \quad 0 \le m < M-1$
	$\beta_m^c(-1) = 0 \quad 0 \le m \le M-1$
	Order initialization
(a)	$\varepsilon_0^f(n) = \varepsilon_0^b(n) = x(n) \quad \varepsilon_0(n) = y(n) \quad \alpha_0(n-1) = 1$
	Lattice part: $m = 0.1, \ldots, M-1$
(b)	$\beta_m(n) = \lambda\beta_m(n-1) + \dfrac{\varepsilon_m^b(n-1)\varepsilon_m^{f*}(n)}{\alpha_m(n-1)}$
(c)	$E_m^f(n) = \lambda E_m^f(n-1) + \dfrac{\|\varepsilon_m^f(n)\|^2}{\alpha_m(n-1)}$
(d)	$E_m^b(n) = \lambda E_m^b(n-1) + \dfrac{\|\varepsilon_m^b(n)\|^2}{\alpha_n(n)}$
(e)	$k_m^f(n) = \dfrac{-\beta_m(n)}{E_m^b(n-1)}$
(f)	$k_m^b(n) = \dfrac{-\beta_m^*(n)}{E_m^f(n)}$
(g)	$\varepsilon_{m+1}^f(n) = \varepsilon_m^f(n) + k_m^{f*}(n)\varepsilon_m^b(n-1)$
(h)	$\varepsilon_{m+1}^b(n) = \varepsilon_m^b(n-1) + k_m^{b*}(n)\varepsilon_m^f(n)$
(i)	$\alpha_{m+1}(n) = \alpha_m(n) - \dfrac{\|a_m(n)e_m^b(n)\|^2}{E_m^b(n)}$
	Ladder part: $m = 1, 2, \ldots, M$
(j)	$\beta_m^c(n) = \lambda\beta_m^c(n-1) + \varepsilon_m^b(n)\varepsilon_m^*(n)/\alpha_m(n)$
(k)	$k_m^c(n) = \dfrac{\beta_m^c(n)}{E_m^b(n)}$
(l)	$\varepsilon_{m+1}(n) = \varepsilon_m(n) - k_m^{c*}(n)\varepsilon_m^b(n)$

If instead of the a posteriori errors we use the a priori ones, we obtain the following recursions

$$e_0^f(n) = e_0^b(n) = x(n)$$

$$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1) \qquad 0 \le m < M-1 \tag{9.6.47}$$

$$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n) \qquad 0 \le m < M-1 \tag{9.6.48}$$

for the lattice part and

$$\begin{aligned} e_0(n) &= y(n) \\ e_{m+1}(n) &= e_m(n) - k_m^{c*}(n-1)e_m^b(n) \qquad 1 \le m \le M \end{aligned} \tag{9.6.49}$$

for the ladder part (see Problem 9.38). As expected, the a priori structure uses the old LS estimates of the lattice-ladder parameters. Based on these recursions, we can develop the a priori error RLS lattice-ladder algorithm

shown in Table 9.13, which requires about 20M operations per time update.

TABLE 9.13
Computational organization of a priori LS lattice-ladder algorithm.

Equation	Computation
	Time initialization
	$E_m^f(-1) = E_m^b(-1) = \delta > 0$
	$\beta_m(-1) = 0 \quad e_m^b(-1) = 0 \quad 0 \le m < M-1$
	$\beta_m^c(-1) = 0 \quad 0 \le m \le M-1$
	Order initialization
(a)	$e_0^f(n) = e_0^b(n) = x(n) \quad e_0(n) = y(n) \quad \alpha_0(n-1) = 1$
	Lattice Part: m = 0.1, ..., M – 2
(b)	$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1)$
(c)	$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n)$
(d)	$\beta_m(n) = \lambda\beta_m(n-1) + \alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)$
(e)	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1)\lvert e_m^f(n)\rvert^2$
(f)	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n)\lvert e_m^b(n)\rvert^2$
(g)	$k_m^f(n) = \dfrac{-\beta_m(n)}{E_m^b(n-1)}$
(h)	$k_m^b(n) = \dfrac{-\beta_m^*(n)}{E_m^f(n)}$
(i)	$\alpha_m(n) = \alpha_{m-1}(n) - \dfrac{\lvert e_{m-1}^b(n)\rvert^2}{E_{m-1}^b(n)}$
	Ladder part: m = 1, 2, ..., M
(j)	$\beta_m^c(n) = \lambda\beta_m^c(n-1) + \alpha_m(n)e_m^b(n)e_m^*(n)$
(k)	$k_m^c(n) = \dfrac{\beta_m^c(n)}{E_m^b(n)}$
(l)	$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1)e_m^b(n)$

9.6.3 *RLS Lattice-Ladder Filters Using Error Feedback Updatings*

The LS lattice-ladder algorithms introduced in the previous section update the partial correlations $\beta_m(n)$ and $\beta_m^c(n)$ and the minimum error energies $E_m^f(n)$ and $E_m^b(n)$, and then compute the coefficients of the LS lattice-ladder filter by division. We next develop two algebraically equivalent algorithms, that is, algorithms that solve the same LS problem, which update the lattice-ladder coefficients *directly*. These algorithms, introduced in Ling et al. (1986), have good numerical properties when implemented with finite-word-length arithmetic.

Starting with (9.6.38) and (9.6.45) we have

$$\begin{aligned} k_m^c(n) &= \frac{\beta_m^c(n)}{E_m^b(n)} = \lambda\frac{\beta_m^c(n-1)}{E_m^b(n-1)}\frac{E_m^b(n-1)}{E_m^b(n)} + \frac{\alpha_m(n)e_m^b(n)e_m^*(n)}{E_m^b(n)} \\ &= \frac{1}{E_m^b(n)}[k_m^c(n-1)\lambda E_m^b(n-1) + \alpha_m(n)e_m^b(n)e_m^*(n)] \end{aligned} \tag{9.6.50}$$

or using
$$\lambda E_m^b(n-1) = E_m^b(n) - \alpha_m(n)e_m^b(n)e_m^{b*}(n)$$

we obtain

$$k_m^c(n) = k_m^c(n-1) + \frac{\alpha_m(n)e_m^{\mathrm{b}}(n)}{E_m^{\mathrm{b}}(n)}[e_m^*(n) - k_m^c(n-1)e_m^{\mathrm{b}*}(n)]$$

or

$$k_m^c(n) = k_m^c(n-1) + \frac{\alpha_m(n)e_m^{\mathrm{b}}(n)e_{m+1}^*(n)}{E_m^{\mathrm{b}}(n)} \tag{9.6.51}$$

using (9.6.49). Equation (9.6.51) provides a *direct* updating of the ladder parameters. Similar direct updating formulas can be obtained for the lattice coefficients (see Problem 9.39). Using these updatings, we obtain the a priori RLS lattice-ladder algorithm with error feedback shown in Table 9.14.

TABLE 9.14
Computational organization of a priori RLS lattice-ladder algorithm with direct updating of its coefficients using error feedback formula.

Equation	Computation
	Time initialization
	$E_m^{\mathrm{f}}(-1) = E_m^{\mathrm{b}}(-1) = \delta > 0$
	$k_m^{\mathrm{f}}(-1) = k_m^{\mathrm{b}}(-1) = 0$
	$e_m^{\mathrm{b}}(-1) = 0 \qquad k_m^c(-1) = 0$
	Order initialization
(a)	$e_0^{\mathrm{f}}(n) = e_0^{\mathrm{b}}(n) = x(n) \qquad e_0(n) = y(n) \qquad \alpha_0(n) = 1$
	Lattice part: *m* = 0.1, ..., *M* − 2
(b)	$e_{m+1}^{\mathrm{f}}(n) = e_m^{\mathrm{f}}(n) + k_m^{\mathrm{f}*}(n-1)e_m^{\mathrm{b}}(n-1)$
(c)	$e_{m+1}^{\mathrm{b}}(n) = e_m^{\mathrm{b}}(n-1) + k_m^{\mathrm{b}*}(n-1)e_m^{\mathrm{f}}(n)$
(d)	$E_m^{\mathrm{f}}(n) = \lambda E_m^{\mathrm{f}}(n-1) + \alpha_m(n-1)\,\lvert e_m^{\mathrm{f}}(n)\rvert^2$
(e)	$E_m^{\mathrm{b}}(n) = \lambda E_m^{\mathrm{b}}(n-1) + \alpha_m(n)\,\lvert e_m^{\mathrm{b}}(n)\rvert^2$
(f)	$k_m^{\mathrm{f}}(n) = k_m^{\mathrm{f}}(n-1) - \dfrac{\alpha_m(n-1)e_m^{\mathrm{b}}(n-1)e_{m+1}^{\mathrm{f}*}(n)}{E_m^{\mathrm{b}}(n-1)}$
(g)	$k_m^{\mathrm{b}}(n) = k_m^{\mathrm{b}}(n-1) - \dfrac{\alpha_m(n-1)e_m^{\mathrm{f}}(n)e_{m+1}^{\mathrm{b}*}(n)}{E_m^{\mathrm{f}}(n)}$
(h)	$\alpha_{m+1}(n) = \alpha_m(n) - \dfrac{\lvert\alpha_m(n)e_m^{\mathrm{b}}(n)\rvert^2}{E_m^{\mathrm{b}}(n)}$
	Ladder part: *m* = 0.1, ..., *M* − 1
(I)	$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1)e_m^{\mathrm{b}}(n)$
(j)	$k_m^c(n) = k_m^c(n-1) + \dfrac{\alpha_m(n)e_m^{\mathrm{b}}(n)e_{m+1}^*(n)}{E_m^{\mathrm{b}}(n)}$

We note that we first use the coefficient $k_m^c(n-1)$ to compute the higher-order error $e_{m+1}(n)$ by (9.6.49) and then use that error to update the coefficient using (9.6.51). This updating has a feedback-like structure that is sometimes referred to as *error feedback form.* An a posteriori form of the RLS lattice-ladder algorithm with error feedback can be easily obtained as shown in Problem 9.40. Simulation studies (Ling et al. 1986) have shown that when we use finite-precision arithmetic, the algorithms with direct updating of the lattice coefficients have better numerical properties than the algorithms with indirect updating.

9.7 Tracking Performance of Adaptive Algorithms

Tracking of a time-varying system is an important problem in many areas of application. Consider, for example, a digital communications system in which the channel characteristics may change with time for various reasons. If we want to incorporate an echo canceler in such a system, then clearly the echo canceler must monitor the changing impulse response of the echo path so that it can generate an accurate replica of the echo. This will require the adaptive algorithm of an echo canceler to possess an acceptable tracking capability. Similar situations arise in adaptive equalization, adaptive prediction, adaptive noise canceling, and so on. In all these applications, adaptive filters are forced to operate in a *nonstationary* SOE. In this section, we examine the ability and performance of the LMS and RLS algorithms to track the ever-changing minimum point of the error surface.

As discussed earlier, the tracking mode is a steady-state operation of the adaptive algorithm, and it follows the acquisition mode, which is a transient phenomenon. Therefore, the algorithm must acquire the system parameters before tracking can commence. This has two implications. First, the rate of convergence is generally not related to the tracking behavior, and as such, we analyze the tracking behavior when the number of iterations (or steps) is relatively large. Second, the time variation of the parameter change should be small enough compared to the rate of convergence that the algorithm can perform adequate tracking; otherwise, it is constantly acquiring the parameters.

9.7.1 Approaches for Nonstationary SOE

To effectively track a nonstationary SOE, adaptive algorithms should use only *local* statistics. There are three practical ways in which this can be achieved.

Exponentially growing window

In this approach, the current data are artificially emphasized by exponentially weighting past data values, as shown in Figure 9.38(*a*). The error function that is minimized is given by

$$E(n)=\sum_{j=0}^{n}\lambda^{n-j}\,|\,y(j)-\mathbf{c}^{\mathrm{H}}\mathbf{x}(j)\,|^2=\lambda E(n-1)+|\,y(n)-\mathbf{c}^{\mathrm{H}}\mathbf{x}(n)\,|^2 \tag{9.7.1}$$

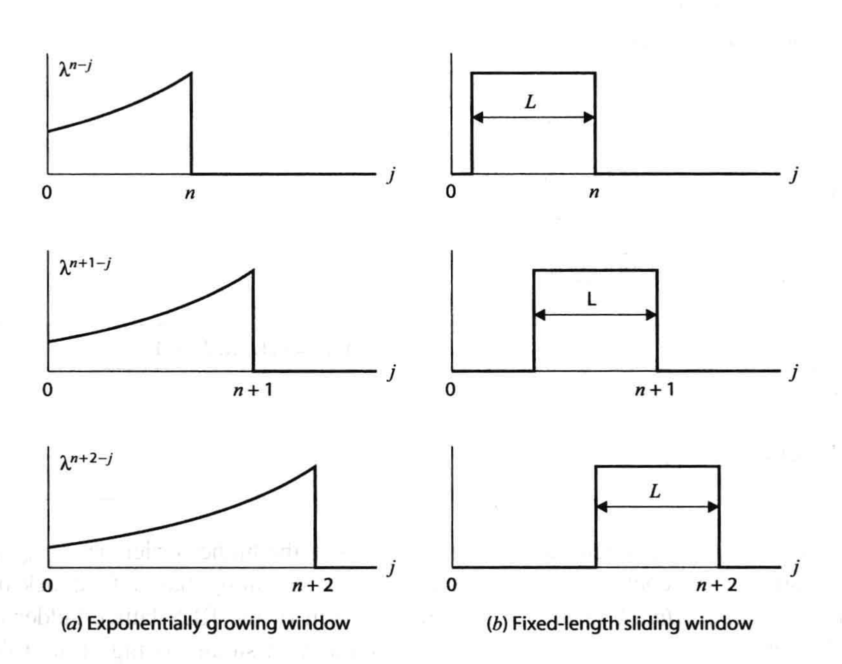


FIGURE 9.38
Illustration of exponentially growing and fixed-length sliding windows.

where $0<\lambda<1$. Clearly, this is the cost function we used in the development of the RLS algorithm, given in Table 9.6, in which λ is termed the forgetting factor. The effective window length is given by

$$L_{\text{eff}} \triangleq \frac{\sum_{n=0}^{\infty} \lambda^n}{\lambda^0} = \frac{1}{1-\lambda} \tag{9.7.2}$$

Hence for good tracking performance λ should be in the range $0.9 \le \lambda < 1$. Note that $\lambda = 1$ results in a *rectangularly growing* window that uses global statistics and hence will not be able to track parameter changes. Thus the RLS algorithm with exponential forgetting is capable of using the local information needed to adapt in a nonstationary SOE.

Fixed-length sliding window

The basic feature of this approach is that the parameter estimates are based only on a finite number of past data values, as shown in Figure 9.38(*b*). Let us consider a rectangular window of fixed length $L > M$. Then the cost function that is minimized is given by

$$E(n,L) \triangleq \sum_{j=n-L+1}^{n} |\, y(j) - \boldsymbol{c}^{\text{H}}\boldsymbol{x}(j)\,|^2 \tag{9.7.3}$$

When a new data value at $n+1$ is added to the sum in (9.7.3), the old data value is discarded, that is, all old data values beyond $n-L+1$ are discarded. Thus the active number of data values is always a constant equal to *L*, which makes this as a constant-memory adaptive algorithm. By following the steps given for the RLS adaptive filter in Section 9.5, it is possible to derive a recursive algorithm to determine the filter $\boldsymbol{c}(n)$ that minimizes the error function in (9.7.3).

Let $\boldsymbol{c}_{\{n-L\}}(n-1)$ denote the estimate of $\boldsymbol{c}(n-1)$ based on *L* data values between $n-L$ and $n-1$. After the new data value at *n* is observed, the RLS algorithm in Table 9.6 is applicable with $\lambda=1$ and with obvious extension of notation. Hence we obtain the algorithm

$$\boldsymbol{c}_{\{n-L\}}(n) = \boldsymbol{c}_{\{n-L\}}(n-1) + \boldsymbol{g}_{\{n-L\}}(n)e^*(n) \tag{9.7.4}$$

$$e(n) = y(n) - \boldsymbol{c}_{\{n-L\}}^{\text{H}}(n-1)\boldsymbol{x}(n) \tag{9.7.5}$$

$$\boldsymbol{g}_{\{n-L\}}(n) = \frac{\bar{\boldsymbol{g}}_{\{n-L\}}(n)}{\alpha_{\{n-L\}}(n)} \tag{9.7.6}$$

$$\bar{\boldsymbol{g}}_{\{n-L\}}(n) = \boldsymbol{P}_{\{n-L\}}(n-1)\boldsymbol{x}(n) \tag{9.7.7}$$

$$\alpha_{\{n-L\}}(n) = 1 + \bar{\boldsymbol{g}}_{\{n-L\}}^{\text{H}}(n)\boldsymbol{x}(n) \tag{9.7.8}$$

$$\boldsymbol{P}_{\{n-L\}}(n) = \boldsymbol{P}_{\{n-L\}}(n-1) - \boldsymbol{g}_{\{n-L\}}(n)\bar{\boldsymbol{g}}_{\{n-L\}}^{\text{H}}(n) \tag{9.7.9}$$

The above algorithm is based on $L+1$ data values. To maintain the data window at fixed length *L*, we have to discard the observation at $n-L$. By using the matrix inversion lemma it can be shown that (see Problem 9.51)

$$\boldsymbol{c}_{\{n-L+1\}}(n) = \boldsymbol{c}_{\{n-L\}}(n) - \boldsymbol{g}_{\{n-L+1\}}(n)e^*(n-L) \tag{9.7.10}$$

$$e(n-L) = y(n-L) - \boldsymbol{c}_{\{n-L\}}^{\text{H}}(n)\boldsymbol{x}(n-L) \tag{9.7.11}$$

$$\boldsymbol{g}_{\{n-L+1\}}(n) = \frac{\bar{\boldsymbol{g}}_{\{n-L+1\}}(n)}{\alpha_{\{n-L+1\}}(n)} \tag{9.7.12}$$

$$\bar{g}_{\{n-L+1\}}(n) = P_{\{n-L\}}(n)x(n-L) \tag{9.7.13}$$

$$\alpha_{\{n-L+1\}}(n) = 1 - \bar{g}^{\mathrm{H}}_{\{n-L+1\}}(n)x(n-L) \tag{9.7.14}$$

$$P_{\{n-L+1\}}(n) = P_{\{n-L\}}(n) + g_{\{n-L+1\}}(n)\bar{g}^{\mathrm{H}}_{\{n-L+1\}}(n) \tag{9.7.15}$$

The overall algorithm for the fixed-memory rectangular window adaptive algorithm is given by (9.7.4) through (9.7.15), which recursively update $\mathbf{c}_{\{n-L\}}(n-1)$ to $\mathbf{c}_{\{n-L+1\}}(n)$. Thus, this algorithm can adapt to the nonstationary SOE using the local information. The fixed-length sliding-window RLS algorithm can be implemented by using a combination of two prewindowed RLS algorithms (Manolakis et al. 1987).

Evolutionary model—Kalman filter

In the first two approaches, adaptation in the nonstationarity SOE was obtained through the local information, either by discarding old data or by deemphasizing it. In the third approach, we assume that we have a statistical model that describes the nonstationarity of the SOE. This model is in the form of a stochastic difference equation together with appropriate statistical properties. This leads to the well-known Kalman filter formulation in which we assume that the parameter variations are modeled by

$$c(n) = \Xi(n)c(n-1) + v(n) \tag{9.7.16}$$

where $v(n)$ is a random vector with zero mean and correlation matrix $\Sigma(n)$, and $\Xi(n)$ is the state-transition matrix known for all n. The desired signal $y(n)$ is modeled as

$$y(n) = c^{\mathrm{H}}(n)x(n) + \varepsilon(n) \tag{9.7.17}$$

where $\varepsilon(n)$ is the a posteriori estimation error assumed to be zero-mean with variance σ_ε^2. Thus in this formulation, the parameter vector $c(n)$ acts as the state of a system while the input data vector $x(n)$ acts as the time-varying output vector. Now the best linear unbiased estimate $\hat{c}(n)$ of $c(n)$ based on past observations $\{y(i)\}_{i=0}^{n}$ can be obtained by using the Kalman filter equations (Section 6.8). These recursive equations are given by

$$\hat{c}(n) = \Xi(n)\hat{c}(n-1) + g(n)[y(n) - \hat{c}^{\mathrm{H}}(n-1)\Xi^{\mathrm{H}}(n)x(n)] \tag{9.7.18}$$

$$g(n) = \frac{\Xi(n)P(n-1)x(n)}{\sigma_\varepsilon^2 + x^{\mathrm{H}}(n)P(n-1)x(n)} \tag{9.7.19}$$

$$\begin{aligned} P(n) &= \Xi(n)P(n-1)\Xi^{\mathrm{H}}(n) + \Sigma(n) \\ &\quad - \Xi(n)P(n-1)\frac{x(n)x^{\mathrm{H}}(n)}{\sigma_\varepsilon^2 + x^{\mathrm{H}}(n)P(n-1)x(n)}P(n-1)\Xi^{\mathrm{H}}(n) \end{aligned} \tag{9.7.20}$$

where $g(n)$ is the Kalman gain matrix and $P(n)$ is the error covariance matrix. This approach implies that if the time-varying parameters are modeled as state equations, then the Kalman filter rather than the adaptive filter is a proper solution.

Furthermore, it can be shown that the Kalman filter has a close similarity to the RLS adpative filters if we make the following appropriate substitutions:

Exponential memory: If we substitute

$$\Xi(n) = I \qquad \sigma_\varepsilon^2 = \lambda \qquad \Sigma(n) = \frac{1-\lambda}{\lambda}[I - g(n)x^{\mathrm{H}}(n)]P(n-1) \tag{9.7.21}$$

then we obtain the exponential memory RLS algorithm given in Table 9.9.

`Rectangularly growing memory`: If we substitute

$$\Xi(n) = \boldsymbol{I} \qquad \sigma_\varepsilon^2 = 1 \qquad \Sigma(n) = 0 \tag{9.7.22}$$

then we obtain the rectangularly growing memory RLS algorithm.

9.7.2 Preliminaries in Performance Analysis

In Sections 9.4 and 9.5.4, we developed and analyzed the LMS and RLS algorithms in stationary environments, respectively. However, these algorithms are generally used in applications (e.g., modems) that are intended to operate continuously in SOE whose characteristics change with time. Therefore, we need to discuss the performance of these two widely used algorithms in such situations. Although we provided various adaptive filtering approaches for time-varying environments above, we now discuss, in the remainder of this section, the ability of these two algorithms to track time-varying parameters. We provide both analytical results, assuming a model of parameter variation, and experimental results, using simulations.

A popular approach for this analytical assessment is to assume a first-order AR model with finite variance [that is we set $\Xi(n) = \rho \boldsymbol{I}$ in (9.7.16)]. Although higher-order models are also possible, only a few results on the tracking performance using these models are currently available. It is ironic that most analytical results on the tracking performance have been obtained for the random-walk model (a special case of the first-order AR model), which is unrealistic because of the infinite variance. A tutorial review of the latest results for the general case and additional references are available in Macchi (1996).

In our analysis of tracking characteristics of the LMS and RLS algorithms, we use the first-order AR model and discuss its effect on the tracking performance. The closed-form results will be given using the random-walk model and confirmed using simulated experiments.

Analysis setup

In the tracking analysis, it is desirable to use the *a priori* adaptive filter. Hence we assume that the desired response is generated by the following filter model[2]

$$y(n) = \boldsymbol{c}_\text{o}^\text{H}(n-1)\boldsymbol{x}(n) + v(n) \tag{9.7.23}$$

where $v(n)$ is assumed to be WGN(0, σ_v^2) with $\sigma_v^2 < \infty$. The random processes $\boldsymbol{x}(n)$ and $v(n)$ are assumed to be independent and stationary. The variation of $\boldsymbol{c}_o(n)$ is modeled by the first-order AR (or Markov) process

$$\boldsymbol{c}_\text{o}(n) = \rho \boldsymbol{c}_\text{o}(n-1) + \boldsymbol{\psi}(n) \tag{9.7.24}$$

with $0 < \rho < 1$ and creates the nonstationarity of the SOE. The quantity $\boldsymbol{\psi}(n)$ is the uncertainty in the model and assumed to be independent of $\boldsymbol{x}(n)$ and $v(n)$, with mean $E\{\boldsymbol{\psi}(n)\} = 0$ and correlation $E\{\boldsymbol{\psi}(n)\boldsymbol{\psi}^\text{H}(n)\} = \boldsymbol{R}_\psi$. Tracking is generally achievable if ρ is close to 1. The random-walk model is obtained by using $\rho = 1$ in (9.7.24).

Conjugate transposing and premultiplying both sides of (9.7.23) by $\mathbf{x}(n)$, taking the expectation, and using independence between $\boldsymbol{x}(n)$ and $v(n)$, we obtain

$$\boldsymbol{R}\boldsymbol{c}_\text{o}(n-1) = \boldsymbol{d}(n) \tag{9.7.25}$$

Hence, $\mathbf{c}_\text{o}(n-1)$ is the optimum a priori filter and

$$e_\text{o}(n) = y(n) - \boldsymbol{c}_\text{o}^\text{H}(n-1)\boldsymbol{x}(n) = v(n) \tag{9.7.26}$$

is the *optimum a priori error*. If $\boldsymbol{R}_\psi = \mathbf{0}$ and $\rho = 1$, we have $\boldsymbol{c}_\text{o}(n) = \boldsymbol{c}_\text{o}$ for all n, and therefore $y(n)$ is wide-sense stationary (WSS). In this case, we have a stationary environment, and the goal of the adaptive filter is to find the optimum filter $\boldsymbol{c}_\text{o}$. For $\boldsymbol{R}_\psi \neq 0$, the adaptive filter should find and track the optimum a priori filter $\boldsymbol{c}_\text{o}(n)$. This setup, which is widely used to analyze the properties of adaptive algorithms, is illustrated in Figure 9.39.

[2]We use this model to make a fair comparison between the adaptive and the optimum filter.

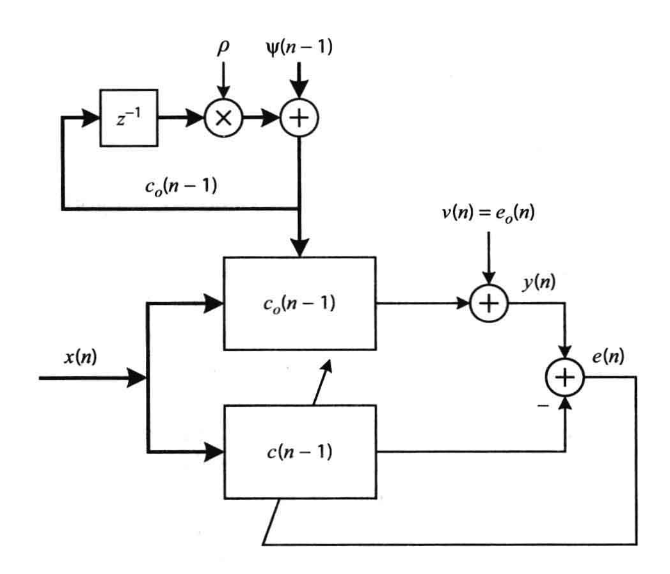


FIGURE 9.39
Block diagram of the setup and model used for the analysis of adaptive algorithms.

Assumptions

To analyze the tracking performance of adaptive algorithms, we use the assumptions discussed elsewhere and repeated below for convenience.

A1 The sequence of input data vectors $\boldsymbol{x}(n)$ is WGN($\mathbf{0}$, $\boldsymbol{R}$).

A2 The desired response $y(n)$ can be modeled as

$$y(n) = \boldsymbol{c}_{\mathrm{o}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n) + e_{\mathrm{o}}(n) \tag{9.7.27}$$

where $e_o(n)$ is WGN$(0, \sigma_o^2)$.

A3 The time variation of $\boldsymbol{c}_{\mathrm{o}}(n)$ is described by

$$\boldsymbol{c}_{\mathrm{o}}(n) = \rho \boldsymbol{c}_{\mathrm{o}}(n-1) + \psi(n) \tag{9.7.28}$$

where $0 \le \rho \le 1$ and $\psi(n)$ is WGN$(0, \boldsymbol{R}_\psi)$.

A4 The random sequences $\boldsymbol{x}(n)$, $e_o(n)$, and $\psi(n)$ are mutually independent.

Through these assumptions, we want to stress that the nonstationarity of the SOE is created solely by $\mathbf{c}_o(n)$ and not by $\mathbf{x}(n)$, which is WSS.

Although we provide analysis for (9.7.27), many results are given for the random walk model ($\rho = 1$). The case $0 < \rho < 1$, which is straightforward but complicated, is discussed in Solo and Kong (1995). Before we delve into this analysis, we discuss criteria that are used for evaluating the tracking performance.

Degree of nonstationarity

To determine whether an adaptive algorithm can adequately track the changing SOE, one needs to define the speed of variation of the statistics of the adaptive filter environment. This speed is quantified in terms of the *degree of nonstationarity* (DNS), introduced in Macchi (1995, 1996), and is defined by

$$\eta(n) \triangleq \sqrt{\frac{E\{|y_{o,\mathrm{incr}}(n)|^2\}}{P_o(n)}} \tag{9.7.29}$$

where

$$y_{\mathrm{o,incr}}(n) = [\boldsymbol{c}_o(n) - \boldsymbol{c}_o(n-1)]^{\mathrm{H}} \boldsymbol{x}(n) \tag{9.7.30}$$

is the output of the *incremental* filter. The numerator is the power introduced by the variation of the optimum filter, and the denominator is the MMSE, which in the context of (9.7.26) is equal to the power of the output noise. Assuming ρ=1 in (9.7.28), we see that (9.7.30) is given by

$$y_{\mathrm{o,incr}}(n) = \boldsymbol{\Psi}^{\mathrm{H}}\,\mathrm{x}(n)$$

and hence the numerator in (9.7.29) is given by

$$\begin{aligned} E\{|\,y_{o,\mathrm{incr}}(n)\,|^2\} &= E\{\Psi^{\mathrm{H}}\mathrm{x}(n)\,\mathrm{x}^{\mathrm{H}}(n)\Psi\} = \mathrm{tr}[E\{\Psi^{\mathrm{H}}\mathrm{x}(n)\,\mathrm{x}^{\mathrm{H}}(n)\Psi\}] \\ &= \mathrm{tr}[E\{\Psi\Psi^{\mathrm{H}}\mathrm{x}(n)\,\mathrm{x}^{\mathrm{H}}\}] = \mathrm{tr}[E\{\Psi\Psi^{\mathrm{H}}\}E\{\mathrm{x}(n)\,\mathrm{x}^{\mathrm{H}}\}\Psi] \\ &= \mathrm{tr}[\mathrm{R}_{\Psi}\mathrm{R}] = \mathrm{tr}[\mathrm{R}\mathrm{R}_{\Psi}] \end{aligned} \tag{9.7.31}$$

where we have used the independence assumption A4. Substituting(9.7.31) in (9.7.29), we obtain

$$\eta(n) \triangleq \sqrt{\frac{\mathrm{tr}[\boldsymbol{R}\boldsymbol{R}_{\Psi}]}{P_{\mathrm{o}}(n)}} \tag{9.7.32}$$

Smaller values of $\eta(<<1)$ imply that the adaptive algorithm can track time variations of the nonstationary SOE. On the contrary, if η>1, then the statistical variations of the SOE are too fast for the adaptive algorithm to keep up with the SOE and lead to massive misadjustment errors. In such situations. an adaptive filter should not be used.

Mean square deviation (MSD)

We defined the MSD $\mathcal{D}(n)$ in (9.2.29) as a performance measure for adaptive filters in the steady-state environment. It is also used for measuring the tracking performance. Consider the coefficient error vector $\tilde{\mathbf{c}}(n)$, which can be written as

$$\begin{aligned} \tilde{\boldsymbol{c}}(n) &= \boldsymbol{c}(n) - \boldsymbol{c}_o(n) \\ &= [\boldsymbol{c}(n) - E\{\boldsymbol{c}(n)\}] + [E\{\boldsymbol{c}(n)\} - \boldsymbol{c}_{\mathrm{o}}(n)] && (9.7.33) \\ &\triangleq \tilde{\boldsymbol{c}}_1(n) + \tilde{\boldsymbol{c}}_2(n) && (9.7.34) \end{aligned}$$

where $\tilde{\boldsymbol{c}}_1(n)$ is the fluctuation of the adaptive filter parameter vector about its mean (estimation error) and $\tilde{\boldsymbol{c}}_2(n)$ is the bias of $\boldsymbol{c}(n)$ with respect to the true vector $\boldsymbol{c}_{\mathrm{o}}(n)$ (systematic or lag error). Using the independence assumption of the previous section that $\boldsymbol{x}(n)$ and $\mathbf{c}(n-1)$ are statistically independent, we can show that (Macchi 1996)

$$E\{\tilde{\boldsymbol{c}}_1^{\mathrm{H}}(n)\tilde{\boldsymbol{c}}_2(n)\} = 0 \tag{9.7.35}$$

which by using (9.2.29) and (9.7.34) leads to

$$\mathcal{D}(n) = \mathcal{D}_1(n) + \mathcal{D}_2(n) \tag{9.7.36}$$

The first MSD term is due to the parameter estimation error and is called the *estimation variance*. The second MSD term is due to the parameter lag error and is termed *lag variance*, and its presence indicates the nonstationary environment.

Misadjustment and lowest excess MSE

The second performance measure, defined in (9.2.38), is the (a priori) misadjustment $\mathcal{M}(n)$, which is the ratio of the excess MSE $P_{\mathrm{ex}}(n)$ to the MMSE $P_o(n)$. The a priori excess MSE is given by

$$P_{\mathrm{ex}}(n) = E\{|\,\tilde{\boldsymbol{c}}^{\mathrm{H}}(n-1)\boldsymbol{x}(n)\,|^2\} = E\{|\,\tilde{\boldsymbol{c}}_1^{\mathrm{H}}(n-1)\boldsymbol{x}(n) + \tilde{\boldsymbol{c}}_2^{\mathrm{H}}(n-1)\boldsymbol{x}(n)\,|^2\} \tag{9.7.37}$$

which under the independence assumption and (9.7.35) can be written as

$$P_{\mathrm{ex}}(n) = P_{\mathrm{ex},1}(n) + P_{\mathrm{ex},2}(n) \tag{9.7.38}$$

where the first term, $P_{\mathrm{ex},1}(n)$, is excess MSE due to estimation error and is termed the *estimation noise* while the second term, $P_{\mathrm{ex},2}(n)$, , is the excess MSE due to lag error and is called the *lag noise*. Therefore, we can also write

the misadjustment $M(n)$ as

$$\mathscr{M}(n) = M_1(n) + M_2(n) \tag{9.7.39}$$

where $\mathscr{M}_1(n)$ is the *estimation misadjustment* and $\mathscr{M}_2(n)$ is the *lag misadjustment.*

In the context of the first-order Markov model, the best performance obtained by any a priori adaptive filter occurs if $\mathbf{c}(n) = \rho \mathbf{c}_o(n-1)$. This observation makes possible the computation of a lower bound for the excess MSE of any a priori adaptive algorithm. From (9.8.34) and (9.8.24), we have

$$\begin{aligned} \tilde{c}(n) &= c(n) - c_o(n) = [c(n) - \rho c_o(n-1)] - \psi(n) \\ &\triangleq \hat{c}(n) - \psi(n) \end{aligned} \tag{9.7.40}$$

and hence

$$P_{ex}(n) = E\{|\tilde{c}^H(n-1)x(n)|^2\}$$

$$= E\{|\hat{c}^{\mathrm{H}}(n-1)x(n) - \psi^{\mathrm{H}}(n-1)x(n)|^2\} \tag{9.7.41}$$

$$\begin{aligned} &= E\{|\hat{c}^{\mathrm{H}}(n-1)x(n)|^2\} + E\{|\psi^{\mathrm{H}}(n-1)x(n)|^2\} \\ &\quad + 2E\{\hat{c}^{\mathrm{H}}(n-1)x(n)x^{\mathrm{H}}(n)\psi(n-1)\} \end{aligned} \tag{9.7.42}$$

Since the term $\hat{\mathbf{c}}(n)$ does not depend on $\psi(n)$ and since the random sequences $x(n)$ and $\psi(n-1)$ are assumed independent, the last term in (9.8.42) is zero. Hence,

$$P_{ex}(n) \geq E\{|\psi^{\mathrm{H}}(n-1)x(n)|^2\} \tag{9.7.43}$$

which provides a lower bound for the excess MSE of any a priori adaptation algorithm. Because $\psi(n)$ and $x(n)$ are assumed independent, we obtain

$$E\{|\psi^{\mathrm{H}}(n-1)x(n)|^2\} = \mathrm{tr}(\boldsymbol{R}\boldsymbol{R}_\psi) \tag{9.7.44}$$

Similarly, neglecting the dependence between $x(n)$ and $\tilde{\mathbf{c}}(n-1)$, we have

$$E\{|\tilde{c}^{\mathrm{H}}(n-1)x(n)|^2\} = \mathrm{tr}[\boldsymbol{R}\boldsymbol{\Phi}(n-1)] \tag{9.7.45}$$

which provides the a priori excess MSE. Furthermore, it can be shown that the DNS places a lower limit on the misadjustment, that is,

$$\mathscr{M}(n) = \frac{P_{ex}(n)}{P_o(n)} \geq \frac{E\{|\psi^{\mathrm{H}}(n-1)x(n)|^2\}}{P_o(n)} = \frac{\mathrm{tr}(\boldsymbol{R}\boldsymbol{R}_\psi)}{\sigma_v^2} = \eta^2(n) \tag{9.7.46}$$

9.7.3 LMS Algorithm

Using the LMS algorithm (9.4.12), the error vector in (9.7.34), and the Markov model in (9.7.28) with $\rho = 1$, we can easily obtain

$$\tilde{c}(n) = [\boldsymbol{I} - 2\mu x(n)x^{\mathrm{H}}(n)]\tilde{c}(n-1) + 2\mu x(n)e_o^*(n) - \psi(n) \tag{9.7.47}$$

which, compared to (9.4.15), has one extra input. Since $x(n)$, $e_o(n)$, and $\psi(n)$ are mutually independent, $\psi(n)$ adds only an extra term $\sigma_\psi^2 \boldsymbol{I}$ to the correlation of $\tilde{\mathbf{c}}(n)$.

Misadjustment. To determine the misadjustment, we perform orthogonal transformation of the correlation matrix of $\tilde{\mathbf{c}}(n)$. When we transform (9.4.28) to (9.4.30), using the orthogonal transformation (9.4.29), the presence of the diagonal matrix $\sigma_\psi^2 \boldsymbol{I}$ changes only the diagonal components with the addition of the term σ_ψ^2. Indeed, we can easily show that

$$\theta_k(n) = \rho_k \theta_k(n-1) + 4\mu^2 \lambda_k P_{\text{ex}}(n-1) + 4\mu^2 P_o \lambda_k + \sigma_\psi^2 \tag{9.7.48}$$

where $P_o(n) = P_o = \sigma_v^2$ for large n. Clearly, (9.7.48) converges under the same conditions as (9.4.40). At steady state we have

$$\theta_k(\infty) = \rho_k \theta_k(\infty) + 4\mu^2 \lambda_k P_{\text{ex}}(\infty) + 4\mu^2 P_o \lambda_k + \sigma_\psi^2 \tag{9.7.49}$$

or using (9.4.36), we have

$$\theta_k(\infty) = \mu \frac{P_o + P_{\text{ex}}(\infty)}{1 - 2\mu\lambda_k} + \frac{1}{4\mu\lambda_k} \frac{\sigma_\psi^2}{1 - 2\mu\lambda_k} \tag{9.7.50}$$

which in conjunction with (9.4.55) and (9.4.56) gives

$$P_{\text{ex}}(\infty) = \frac{C(\mu)}{1 - C(\mu)} \sigma_v^2 + \frac{1}{4\mu} \frac{D(\mu)}{1 - C(\mu)} \sigma_\psi^2 \tag{9.7.51}$$

where

$$D(\mu) \triangleq \sum_{k=1}^{M} \frac{1}{1 - 2\mu\lambda_k} \tag{9.7.52}$$

If $\mu\lambda_k \ll 1$, we have $C(\mu) \simeq \mu \operatorname{tr}(\boldsymbol{R})$ and $D(\mu) \simeq M$, which lead to

$$P_{\text{ex}}(\infty) \simeq \mu \sigma_v^2 \operatorname{tr}(\boldsymbol{R}) + \frac{1}{4\mu} M \sigma_\psi^2 \tag{9.7.53}$$

or

$$\mathcal{M}(\infty) \simeq \mu \operatorname{tr}(\boldsymbol{R}) + \frac{1}{4\mu} M \frac{\sigma_\psi^2}{\sigma_v^2} \tag{9.7.54}$$

Hence in the steady state, the misadjustment can be approximated by two terms. The first term is estimation misadjustment, which increases with μ, while the second term is the lag misadjustment, which decreases with μ. Therefore, an optimum value of μ exists that minimizes $\mathcal{M}(\infty)$, given by

$$\mu_{\text{opt}} \simeq \frac{\sigma_\psi}{2\sigma_v} \sqrt{\frac{M}{\operatorname{tr}(\mathbf{R})}} \tag{9.7.55}$$

or

$$\mathcal{M}_{\text{min}}(\infty) \simeq \frac{\sigma_\psi}{\sigma_v} \sqrt{M \operatorname{tr}(\mathbf{R})} \tag{9.7.56}$$

MSD. To determine the MSD, consider (9.7.47). For small step size μ, the system matrix $[\boldsymbol{I} - 2\mu \boldsymbol{x}(n)\boldsymbol{x}^H(n)]$ is very close to the identity matrix. Hence using the direct averaging method due to Kushner (1984), we can obtain a close solution of $\tilde{c}(n)$ by solving (9.8.47) in which the system matrix is replaced by its average $[\boldsymbol{I} - 2\mu\boldsymbol{R}]$, that is,

$$\tilde{\boldsymbol{c}}(n) = [\boldsymbol{I} - 2\mu\boldsymbol{R}]\tilde{\boldsymbol{c}}(n-1) + 2\mu\boldsymbol{x}(n)e_o^*(n) - \psi(n) \tag{9.7.57}$$

where we have kept the same notation. Taking the covariance of both sides of (9.7.57), we obtain

$$\boldsymbol{\Phi}(n) = [\boldsymbol{I} - 2\mu\boldsymbol{R}]\boldsymbol{\Phi}(n-1)[\boldsymbol{I} - 2\mu\boldsymbol{R}] + 4\mu^2\sigma_v^2\boldsymbol{R} + \boldsymbol{R}_\psi \tag{9.7.58}$$

The approximate steady-state solution of (9.7.58) is given by

$$\boldsymbol{R\Phi} + \boldsymbol{\Phi R} \simeq 2\mu\sigma_v^2\boldsymbol{R} + \frac{\boldsymbol{R}_\psi}{2\mu} \tag{9.7.59}$$

where the second-order term $4\mu^2\boldsymbol{R\Phi R}$ is ignored for small values of μ. After premultiplying (9.7.59) by $\boldsymbol{R}^{-1}$, we obtain

$$\boldsymbol{\Phi} + \boldsymbol{R}^{-1}\boldsymbol{\Phi}\boldsymbol{R} \simeq 2\mu\sigma_v^2 + \frac{\boldsymbol{R}^{-1}\boldsymbol{R}_\psi}{2\mu} \tag{9.7.60}$$

Taking the trace of (9.7.60) and using $tr(\boldsymbol{R}^{-1}\boldsymbol{\Phi}\boldsymbol{R}) = \text{tr}(\boldsymbol{\Phi})$, we obtain

$$\text{tr}(\boldsymbol{\Phi}) \simeq \mu M\sigma_v^2 + \frac{\text{tr}(\boldsymbol{R}^{-1}\boldsymbol{R}_\psi)}{4\mu} \tag{9.7.61}$$

By following the development in (9.7.28), it can be shown that (Problem 9.52) $\mathcal{D}(\infty) = tr(\Phi)$. Hence

$$\mathcal{D}(\infty) \simeq \mu M\sigma_v^2 + \frac{\text{tr}(\boldsymbol{R}^{-1}\boldsymbol{R}_\psi)}{4\mu} \tag{9.7.62}$$

As expected, the MSD has two terms: The estimation deviation is linearly proportional to μ while the lag deviation is inversely proportional to μ. The optimum value of the step size μ is obtained when both deviations are equal and is given by

$$\mu_{\text{opt}} \simeq \frac{1}{2}\sqrt{\frac{\text{tr}(\boldsymbol{R}^{-1}\boldsymbol{R}_\psi)}{M\sigma_v^2}} \tag{9.7.63}$$

or

$$\mathcal{D}_{\text{min}}(\infty) = \sqrt{M\sigma_v^2\text{tr}(\boldsymbol{R}^{-1}\boldsymbol{R}_\psi)} \tag{9.7.64}$$

EXAMPLE 9.7.1. To study the tracking performance of the LMS algorithm, we will simulate a slowly time-varying SOE whose parameters follow an almost random-walk behavior. The simulation setup is shown in Figure 9.42 and given by (9.7.27) and (9.7.28). The simulation parameters are as follows:

$$\mathbf{c}_o(n) \text{ model parameters}: \quad \mathbf{c}_o(0) = \begin{bmatrix} -0.8 \\ 0.95 \end{bmatrix} \qquad M = 2 \qquad \rho = 0.999$$

$$\psi(n) \sim \text{WGN}(\boldsymbol{0},\ \boldsymbol{R}_\psi) \qquad \boldsymbol{R}_\psi = (0.01)^2\boldsymbol{I}$$

$$\text{Signal } \boldsymbol{x}(n) \text{ parameters}: \quad \boldsymbol{x}(n) \sim \text{WGN}(\boldsymbol{0},\ \boldsymbol{R}) \qquad \boldsymbol{R} = \boldsymbol{I}$$

$$\text{Noise } v(n) \text{ parameters}: \quad v(n) \sim \text{WGN}(0,\ \sigma_v^2) \qquad \sigma_v = 0.1$$

For these values, the degree of nonstationarity from (9.7.32) is given by

$$\eta(n) = \frac{\sqrt{\text{tr}[\boldsymbol{R}\boldsymbol{R}_\psi]}}{\sigma_v} = 0.1414 < 1$$

which means that the LMS can track the time variations of the SOE.

Three different adaptations (slow, matched, and fast) of the LMS algorithm were designed. Their adaptation results are shown in Figures 9.40 through 9.45. From (9.7.55) and (9.7.63), the optimum performance is obtained when

$$\mu_{\text{opt}} = 0.05$$

for which $\mathcal{M}_{\text{min}}(\infty) = 0.2$ and $\mathcal{D}_{\text{min}}(\infty) = 0.002$. Hence, the following values for μ were selected for simulation:

$$\text{Slow}: \quad \mu = 0.01$$

$$\text{Matched}: \quad \mu = 0.1$$

$$\text{Fast}: \quad \mu = 0.3$$

Figure 9.40 shows the matched adaptation of parameter coefficients while Figure 9.41 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. Clearly, the LMS tracks the varying coefficients nicely with expected small misregistration and deviation errors. Figure 9.42 shows the slow adaptation of parameter coefficients while Figure 9.43 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the LMS algorithm tracks with bounded error variance, the tracking is not very good and the resulting misregistration errors are large. Finally, Figure 9.44 shows the fast adaptation of parameter coefficients while Figure 9.45 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the algorithm is able to keep track of the slowly varying coefficients, the resulting variance is large and hence the estimation errors are large. Once again, the total errors are large compared to those for the matched case.

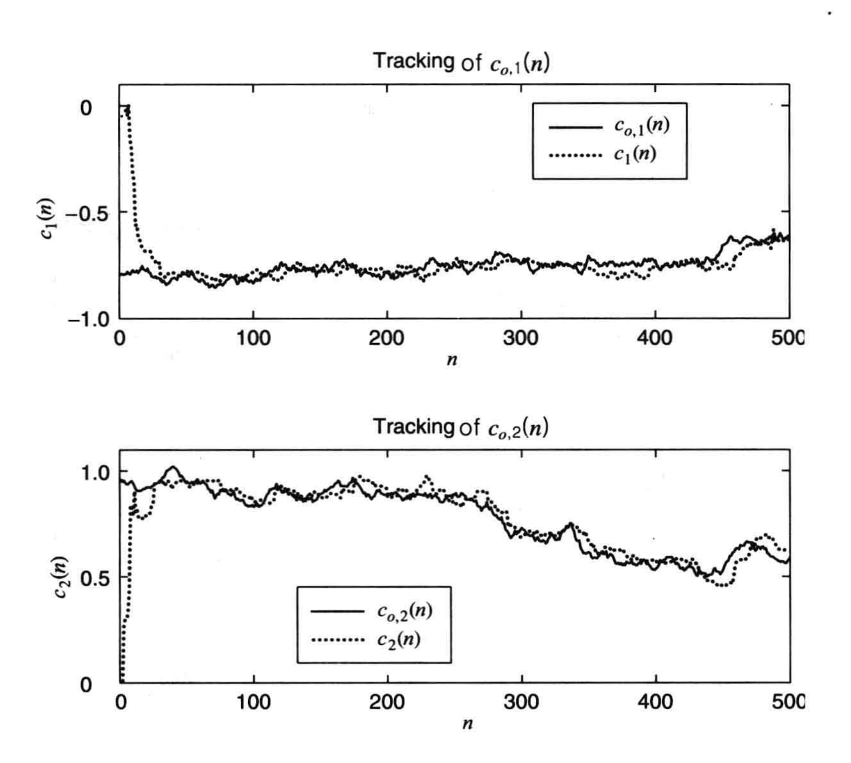


FIGURE 9.40
Matched adaptation of slowly time-varying parameters: LMS algorithm with μ=0.1

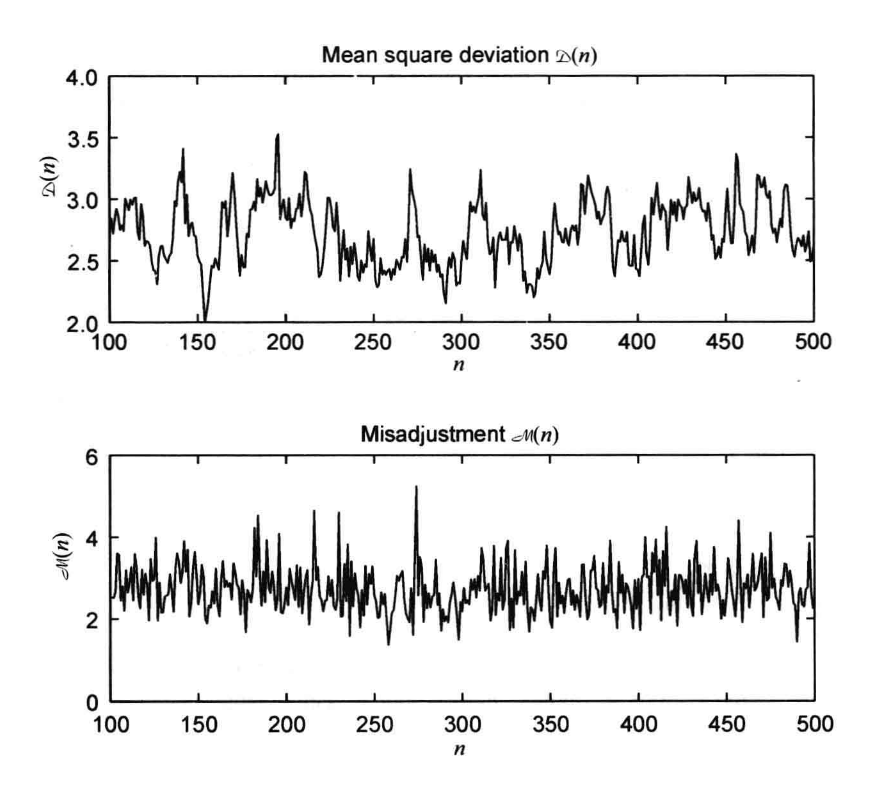


FIGURE 9.41
Learning curves of LMS algorithm with matched adaptation.

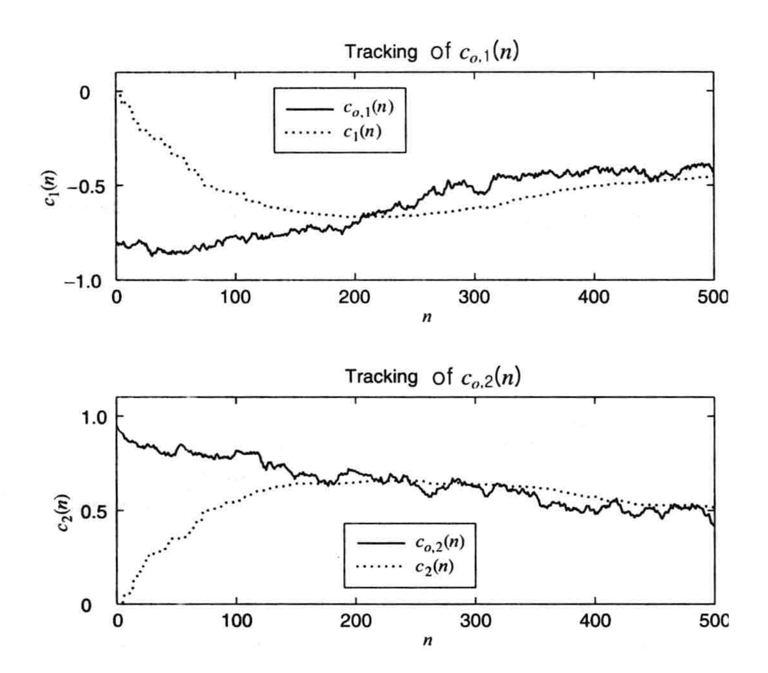


FIGURE 9.42
Slow adaptation of slowly time-varying parameters: LMS algorithm with $\mu = 0.01$

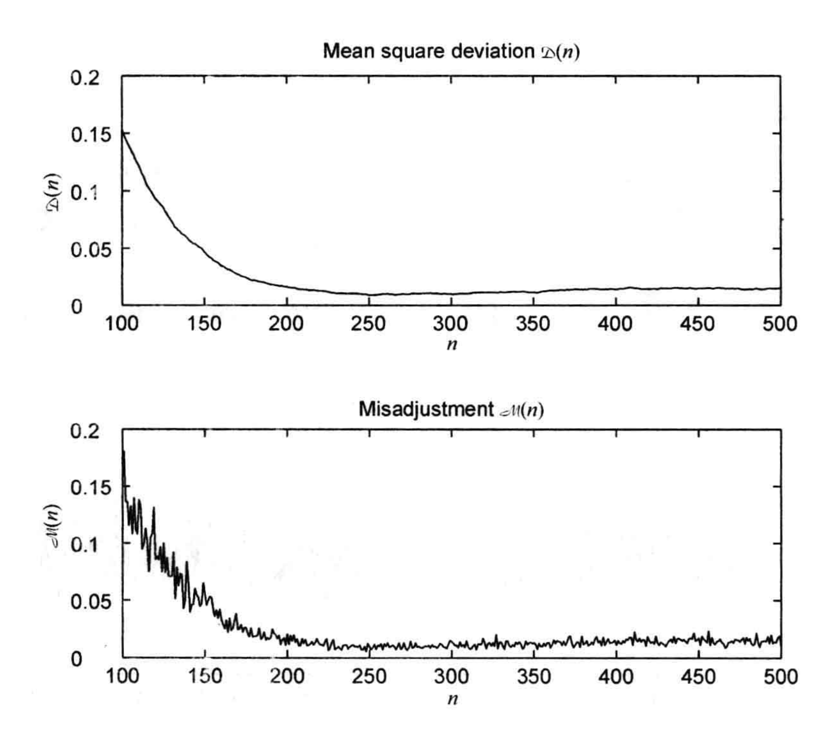


FIGURE 9.43
Learning curves of LMS algorithm for slow adaptation.

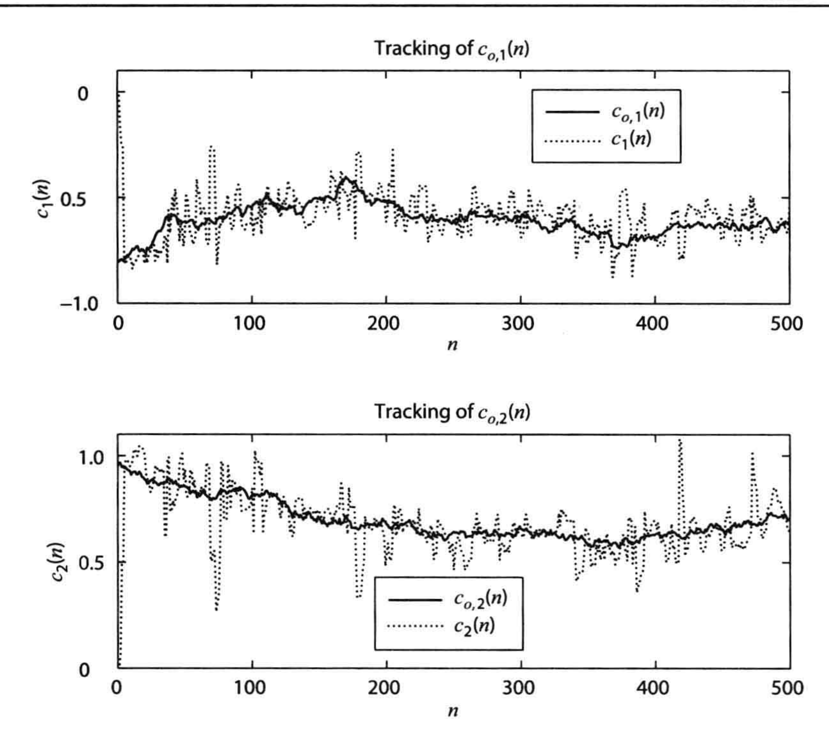


FIGURE 9.44
Fast adaptation of slowly time-varying parameters: LMS algorithm with $\mu = 0.3$.

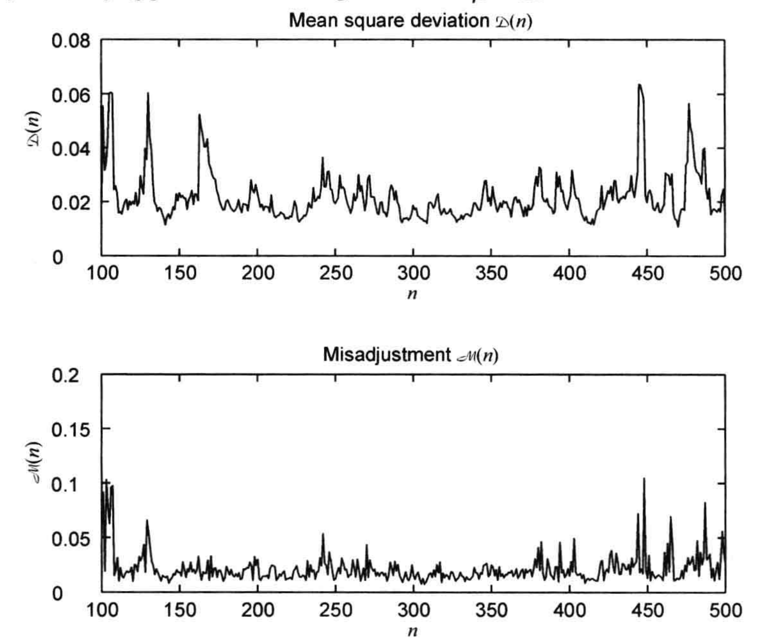


FIGURE 9.45
Learning curves of LMS algorithm for fast adaptation.

9.7.4 RLS Algorithm with Exponential Forgetting

Consider again the model given in Figure 9.40 and described in the analysis setup.

Misadjustment. To determine the misadjustment in tracking, we first evaluate the excess MSE caused by lag, that is, by the deviation between $E\{\mathbf{c}(n)\}$ and the optimum a priori filter $\boldsymbol{c}_o(n)$. Combining

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)e^*(n) \tag{9.7.65}$$

with $$e^*(n) = e_o^*(n) - \boldsymbol{x}^H(n)[\boldsymbol{c}(n-1) - \boldsymbol{c}_o(n-1)] \tag{9.7.66}$$

and taking the expectation result in

$$E\{\boldsymbol{c}(n)\} = E\{\boldsymbol{c}(n-1)\} + E\{\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\}[E\{\boldsymbol{c}(n-1)\} - \boldsymbol{c}_o(n-1)] \tag{9.7.67}$$

because the expectation of $\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)e_o^*(n)$ vanishes. Using the approximation $E\{\hat{\boldsymbol{R}}^{-1}(n)\cdot\boldsymbol{x}(n)\ \mathbf{x}^H(n)\} \simeq (1-\lambda)\mathbf{I}$, we have

$$\tilde{\boldsymbol{c}}_{\text{lag}}(n) \simeq \lambda\tilde{\boldsymbol{c}}_{\text{lag}}(n) + \boldsymbol{c}_o(n-1) - \boldsymbol{c}_o(n) \tag{9.7.68}$$

or $$\tilde{\boldsymbol{c}}_{\text{lag}}(n) \simeq \lambda\tilde{\boldsymbol{c}}_{\text{lag}}(n-1) - \boldsymbol{\psi}(n) \tag{9.7.69}$$

for the random-walk ($\rho = 1$) model. The covariance matrix is

$$\boldsymbol{\Phi}_{\text{lag}}(n) \simeq \lambda^2\boldsymbol{\Phi}_{\text{lag}}(n-1) + \boldsymbol{R}_\psi \tag{9.7.70}$$

and in steady state (assuming $0 < \lambda < 1$)

$$\boldsymbol{\Phi}_{\text{lag}}(\infty) \simeq \frac{1}{(1-\lambda)^2}\boldsymbol{R}_\psi \tag{9.7.71}$$

The lag excess MSE is

$$P_{\text{lag}}(\infty) = \text{tr}[\boldsymbol{R}\boldsymbol{\Phi}(\infty)] \simeq \frac{1}{(1-\lambda)^2}\text{tr}[\boldsymbol{R}\boldsymbol{R}_\psi] \simeq \frac{1}{2(1-\lambda)}\text{tr}[\boldsymbol{R}\boldsymbol{R}_\psi] \tag{9.7.72}$$

because $(1-\lambda)^2 = (1+\lambda)(1-\lambda) \simeq 2(1-\lambda)$ for $\lambda \simeq 1$.
The excess MSE due to estimation is $[(1-\lambda)/2]M\sigma_\nu^2$, hence the total excess MSE is

$$P_{\text{ex}}(\infty) \simeq \frac{1-\lambda}{2}M\sigma_\nu^2 + \frac{1}{2(1-\lambda)}\sigma_\psi^2\text{tr}(\boldsymbol{R}) \tag{9.7.73}$$

if $\boldsymbol{R}_\psi = \sigma_\psi^2\boldsymbol{I}$. Finally, the misadjustment is given by

$$\mathscr{M}(\infty) \simeq \frac{1-\lambda}{2}M + \frac{\sigma_\psi^2\,\text{tr}(\boldsymbol{R})}{2(1-\lambda)\sigma_\nu^2} \tag{9.7.74}$$

The first term in (9.7.74) is the estimation misadjustment, which is linearly proportional to $1-\lambda$, while the second term is the lag misadjustment, which is inversely proportional to $1-\lambda$. The optimum value of λ is given by

$$\lambda_{\text{opt}} \simeq 1 - \frac{\sigma_\psi}{\sigma_\nu}\sqrt{\frac{1}{M}\text{tr}(\boldsymbol{R})} \tag{9.7.75}$$

and the minimum misadjustment is given by

$$\mathscr{M}_{\text{min}}(\infty) \simeq \frac{\sigma_\psi}{\sigma_\nu}\sqrt{M\ \text{tr}(\boldsymbol{R})} \tag{9.7.76}$$

MSD. An analysis similar to the MSD development of the LMS algorithm can be done to obtain

$$D(\infty) \simeq \frac{1-\lambda}{2}\sigma_\nu^2\text{tr}(\boldsymbol{R}^{-1}) + \frac{\sigma_\psi^2}{2(1-\lambda)} \tag{9.7.77}$$

with $$\lambda_{\text{opt}} \simeq 1 - \frac{\sigma_\psi}{\sigma_\nu}\sqrt{\frac{1}{\text{tr}(\mathbf{R}^{-1})}} \tag{9.7.78}$$

and
$$\mathscr{D}_{\min}(\infty) \simeq \frac{\sigma_\psi \sigma_v}{2}\sqrt{\operatorname{tr}(\boldsymbol{R}^{-1})} \tag{9.7.79}$$

which again highlights the dependence of tracking abilities on λ.

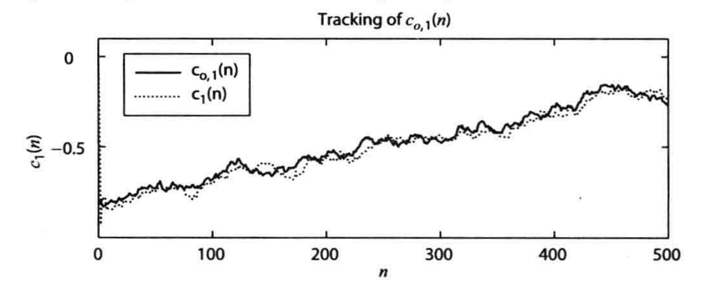
EXAMPLE 9.7.2. To study the tracking performance of the RLS algorithm, we again simulate the slowly time-varying SOE given in Example 9.7.1 whose parameters are repeated here:

$$\mathbf{c}_o(n) \text{ model parameters}: \quad \boldsymbol{c}_o(0) = \begin{bmatrix} -0.8 \\ 0.95 \end{bmatrix} \quad M = 2 \quad \rho = 0.999$$
$$\psi(n) \sim \text{WGN}(\boldsymbol{0}, \boldsymbol{R}_\psi) \quad \boldsymbol{R}_\psi = (0.01)^2 \boldsymbol{I}$$
$$\text{Signal } \boldsymbol{x}(n) \text{ parameters}: \quad \boldsymbol{x}(n) \sim \text{WGN}(\mathbf{0}, \boldsymbol{R}) \quad \boldsymbol{R} = \boldsymbol{I}$$
$$\text{Noise } v(n) \text{ parameters}: \quad v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$$

For these values, the degree of nonstationarity is $\eta(n) = 0.1414$, which means that the RLS can track the time variations of the SOE.

Three different adaptations (slow, matched, and fast) of the RLS algorithm were designed. Their adaptation results are shown in Figures 9.46 through 9.51. From (9.7.75) and (9.7.77), the optimum misadjustment performance is obtained when
$$\lambda_{opt} = 0.9 \qquad \text{with } \mathscr{M}_{\min}(\infty) = 0.2$$
while from (9.7.78) and (9.7.79), the optimum deviation performance is obtained when
$$\lambda_{opt} = 0.93 \qquad \text{with } \mathscr{D}_{\min}(\infty) = 0.007$$
Hence, the following values for λ were selected for simulation:
$$\text{Slow}: \quad \lambda = 0.99$$
$$\text{Matched}: \quad \lambda = 0.9$$
$$\text{Fast}: \quad \lambda = 0.5$$

Figure 9.46 shows the matched adaptation of parameter coefficients while Figure 9.47 shows the resulting $\mathscr{D}(n)$ and $\mathscr{M}(n)$. Clearly, the RLS tracks the varying coefficients nicely with expected small misregistration and deviation errors. Figure 9.48 shows the slow adaptation of parameter coefficients while Figure 9.49 shows the resulting $\mathscr{D}(n)$ and $\mathscr{M}(n)$. In this case, although the RLS algorithm tracks with bounded error variance, the tracking is not very good and the resulting misregistration errors are large. Finally, Figure 9.50 shows the fast adaptation of parameter coefficients while Figure 9.51 shows the resulting $\mathscr{D}(n)$ and $\mathscr{M}(n)$. In this case, although the algorithm is able to keep track of the slowly varying coefficients, the resulting variance is large and hence the estimation errors are large. Once again, the total errors are large compared to those for the matched case.



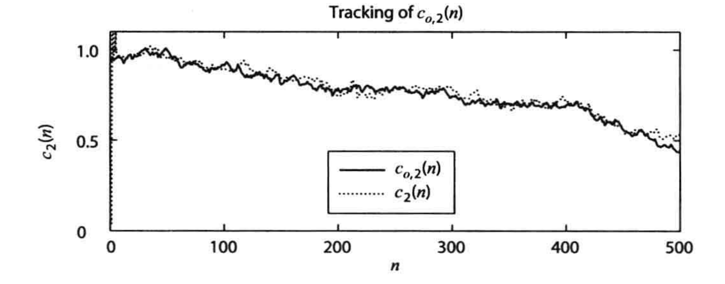


FIGURE 9.46
Matched adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.9$.

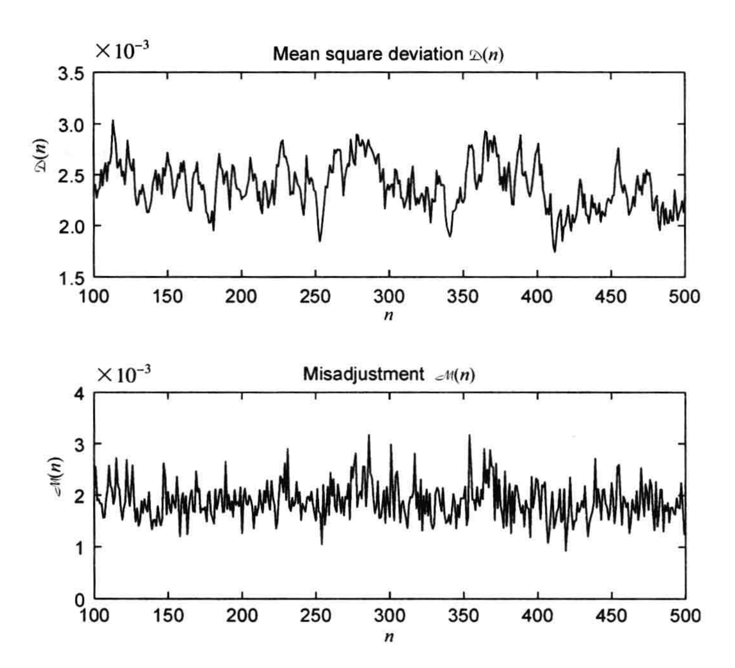


FIGURE 9.47
Learning curves of RLS algorithm for matched adaptation.

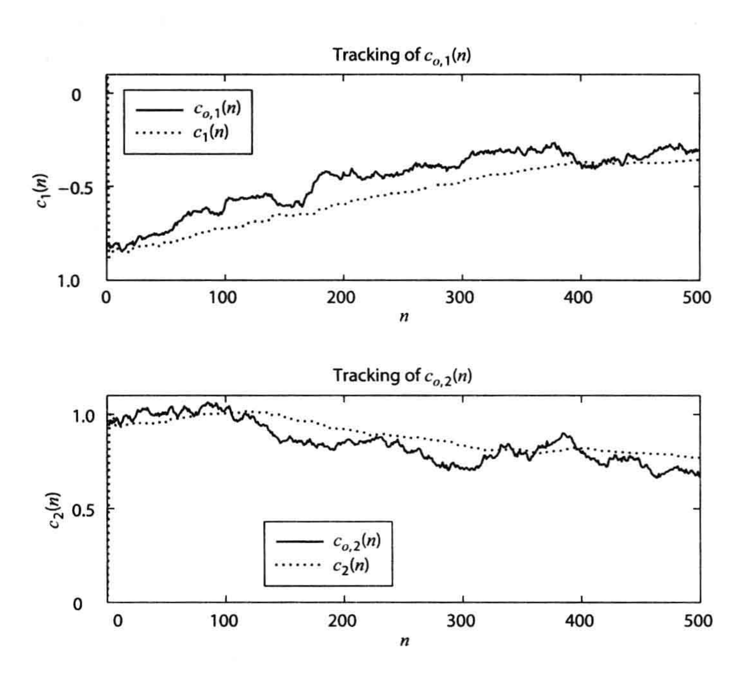


FIGURE 9.48
Slow adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.99$.

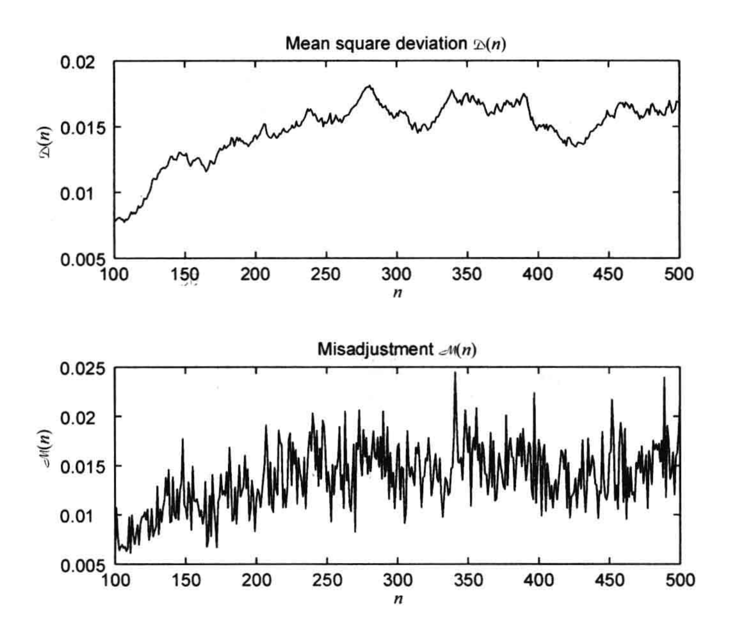


FIGURE 9.49
Learning curves of RLS algorithm for slow adaptation.

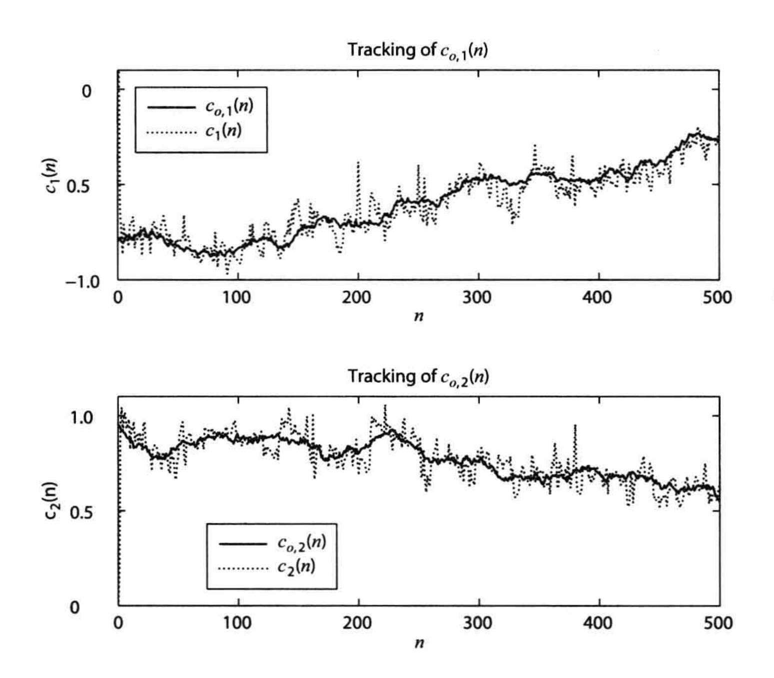


FIGURE 9.50
Fast adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.5$.

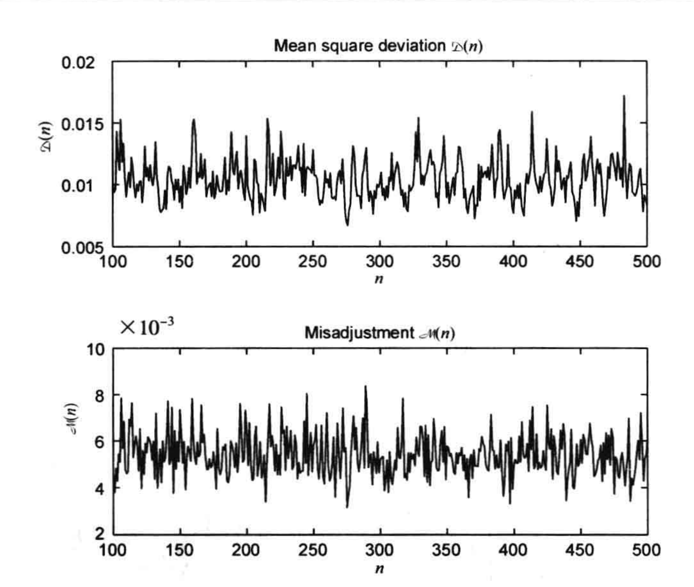


FIGURE 9.51
Learning curves of RLS algorithm for fast adaptation.

9.7.5 Comparison of Tracking Performance

When the optimum filter drifts like a random walk with small increment variance σ_ψ^2, the tracking performance for the LMS algorithm is given by (9.7.54) and (9.7.62) while that for the RLS algorithm is given by (9.7.74) and (9.7.77). Whether the LMS or the RLS algorithm is better depends on matrices $\boldsymbol{R}$ and $\boldsymbol{R}_\psi$. A general comparison is difficult to make, but some guidelines have been developed for particular cases. It has been shown that (Haykin 1996)

- When $\boldsymbol{R}_\psi = \sigma_\psi^2 \boldsymbol{I}$, then both the LMS and RLS algorithms produce essentially the same minimum levels of MSD and misadjustment. However, this analysis is true only asymptotically and for slowly varying parameters (small σ_ψ^2).
- When $\boldsymbol{R}_\psi = \alpha \boldsymbol{R}$ where α is a constant, then the LMS algorithm produces smaller values of the minimum levels of MSD and misadjustment than the RLS algorithm does.
- When $\boldsymbol{R}_\psi = \beta \boldsymbol{R}^{-1}$ where β is a constant, then the RLS algorithm is better than the LMS algorithm in producing the smaller values of the minimum levels of MSD and misadjustment.

In summary, we should state that in practice the comparison of the acquisition and tracking performance of LMS and RLS adaptive filters is a very complicated subject. Although the previous analysis provides some insight only extensive simulations in the context of a specific application can help to choose the appropriate algorithm.

9.8 Summary

In this chapter we discussed the theory of operation, design, performance evaluation, implementation, and applications of adaptive filters. The most significant attribute of an adaptive filter is its ability to incrementally adjust its coefficients so as to improve a predefined criterion of performance over time.

We basically developed and analyzed two families of adaptive filtering algorithms:

- The family of LMS FIR adaptive filters, which are based on a stochastic version of the steepest-descent optimization algorithm.
- The family of RLS FIR adaptive filters, which are based on a stochastic version of the Newton-type optimization algorithms.

Both types of approaches can be used to develop adaptive algorithms for direct-form and lattice-ladder FIR filter

structures.

For LMS adaptive filters we focused on direct-form structures because those are the most widely used and studied. However, we briefly discussed transform-domain and subband implementations because they offer a viable solution for applications that require adaptive filters with very long impulse responses.

All RLS FIR adaptive filters discussed in this chapter exhibit identical performance if they are implemented using infinite-precision arithmetic. The LMS algorithm (Section 9.4), the CRLS algorithm (Section 9.5), the fast RLS algorithms in Section 9.6 can be used only for FIR filtering and prediction applications. The steady-state performance of LMS and RLS algorithms in a stationary environment is discussed in Sections 9.4 and 9.5, whereas their tracking performance in a nonstationary environment is analyzed in Section 9.7.

The treatment of adaptive filters in this chapter has been quite extensive, in both number of topics and depth. However, the following important topics have been omitted:

- IIR adaptive filters (Treichler et al. 1987; Johnson 1984; Shynk 1989; Regalia 1995; Netto et al. 1995; Williamson 1998). Although adaptive IIR filters have the potential to offer the same performance as FIR filters with less computational complexity, they are not widely used in practical applications. The main reasons are related to the nonquadratic nature of their performance error surface (see Section 5.2) and the additional stability problems caused by the presence of poles in their system function.
- Adaptive filters using nonlinear filtering structures and neural networks (Grant and Mulgrew 1995; Haykin 1996; Mathews 1991). The need for such filters arises in applications involving nonlinear input-output relationships, nonlinear detectors (e.g., data equalization), and non-Gaussian or impulsive noise. The optimization required in some of these cases can be performed using genetic optimization algorithms (Tang et al. 1996).
- FIR direct-form and lattice-ladder LS adaptive filters for multichannel signals (Slock 1993; Ling 1993b; Carayannis et al. 1986).

Problems

9.1 Consider the process $x(n)$ generated using the AR(3) model

$$x(n) = -0.729x(n-3) + \omega(n)$$

where $\omega(n) \sim \text{WGN}(0,1)$. We want to design a linear predictor of $x(n)$ using the SDA algorithm. Let

$$\hat{y}(n) = \hat{x}(n) = c_{o,1}x(n-1) + c_{o,2}x(n-2) + c_{o,3}x(n-3)$$

(*a*) Determine the 3×3 autocorrelation matrix $\boldsymbol{R}$ of $x(n)$, and compute its eigenvalues $\{\lambda_i\}_{i=1}^{3}$.

(*b*) Determine the 3×1 cross-correlation vector $\boldsymbol{d}$.

(*c*) Choose the step size μ so that the resulting response is overdamped. Now implement the SDA

$$\boldsymbol{c}_k = [c_{k,1}\ c_{k,2}\ c_{k,3}]^{\mathrm{T}} = \boldsymbol{c}_{k-1} + 2\mu(\boldsymbol{d} - \boldsymbol{R}\boldsymbol{c}_{k-1})$$

and plot the trajectories of $\{c_{k,i}\}_{i=1}^{3}$ as a function of *k*.

(*d*) Repeat part (*c*) by choosing μ so that the response is underdamped.

9.2 In the SDA algorithm, the index *k* is an iteration index and not a time index. However, we can treat it as a time index and use the instantaneous filter coefficient vector $\mathbf{c}_k$ to filter data at $n = k$. This will result in an *asymptotically optimum* filter whose coefficents will converge to the optimum one. Consider the process $x(n)$ given in Problem 9.1.

(a) Generate 500 samples of $x(n)$ and implement the asymptotically optimum filter. Plot the signal $\hat{y}(n)$.

(b) Implement the optimum filter $\mathbf{c}_o$ on the same sequence, and plot the resulting $\hat{y}(n)$.

(c) Comment on the above two plots.

9.3 Consider the AR(2) process $x(n)$ given in Example 9.3.1. We want to implement the Newton-type algorithm for faster convergence using

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} - \mu\boldsymbol{R}^{-1}\nabla P(\boldsymbol{c}_{k-1})$$

(a) Using $a_1 = -1.5955$ and $a_2 = 0.95$, implement the above method for $\mu = 0.1$ and $\mathbf{c}_0 = 0$. Plot the locus of $c_{k,1}$ versus $c_{k,2}$.

(b) Repeat part (*a*), using $a_1 = -0.195$ and $a_2 = 0.95$.

(c) Repeat parts (*a*) and (*b*), using the optimum step size for μ that results in the fastest convergence.

9.4 Consider the adaptive linear prediction of an AR(2) process $x(n)$ using the LMS algorithm in which

$$x(n) = 0.95x(n-1) - 0.9x(n-2) + \omega(n)$$

where $\omega(n) \sim \text{WGN}(0,\ \sigma_\omega^2)$. The adaptive predictor is a second-order one given by $\boldsymbol{a}(n) = [a_1(n)\ a_2(n)]^T$.

(a) Implement the LMS algorithm given in Table 9.3 as a MATLAB function

```
[a,e]=lplms(x,y,mu,M,a0)
```

which computes filter coefficients in `c` and the corresponding error in `e`, given signal `x`, desired signal `y`, step size `mu`, filter order `M`, and the initial coefficient vector `a0`.

(b) Generate 500 samples of $x(n)$, and obtain linear predictor coefficients using the above function. Use step size μ so that the algorithm converges in the mean. Plot predictor coefficients as a function of time along with the true coefficients.

(c) Repeat the above simulation 1000 times to obtain the learning curve, which is obtained by averaging the squared error $|e(n)|^2$. Plot this curve and compare its steady-state value with the theoretical MSE.

9.5 Consider the adaptive echo canceler given in Figure 9.25. The FIR filter $c_o(n)$ is given by

$$c_o(n) = (0.9)^n \qquad 0 \le n \le 2$$

In this simulation, ignore the far-end signal $u(n)$. The data signal $x(n)$ is a zero-mean, unit-variance white Gaussian process, and $y(n)$ is its echo.

(a) Generate 1000 samples of $x(n)$ and determine $y(n)$. Use these signals to obtain a fourth-order LMS echo canceler in which the step size μ is chosen to satisfy (9.4.40) and $\boldsymbol{c}(0) = 0$. Obtain the final echo canceler coefficients and compare them with the true ones.

(b) Repeat the above simulation 500 times, and obtain the learning curve. Plot this curve along with the actual MSE and comment on the plot.

(c) Repeat parts (*a*) and (*b*), using a third-order echo canceler.

(d) Repeat parts (*a*) and (*b*), using one-half the value of μ used in the first part.

9.6 The normalized LMS (NLMS) algorithm is given in (9.4.67), in which the effective step size is time-varying and is given by $\tilde{\mu}/\|\boldsymbol{x}(n)\|^2$, where $0 < \tilde{\mu} < 1$.

(a) Modify the function `firlms` to implement the NLMS algorithm and obtain the function

```
[c,e]=nfirlms(x, y, mu, M, c0)
```

(b) Choose $\tilde{\mu} = 0.1$ and repeat Problem 9.4. Compare your results in terms of convergence speed.

(c) Choose $\tilde{\mu} = 0.1$ and repeat Problem 9.5(*a*) and (*b*). Compare your results in terms of convergence speed.

9.7 Another variation of the LMS algorithm is called the *sign-error* LMS algorithm, in which the coefficient update equation is given by

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + 2\mu\,\text{sgn}[e(n)]\boldsymbol{x}(n)$$

where

$$\text{sgn}[e(n)] = \begin{cases} 1 & \text{Re}[e(n)] > 0 \\ 0 & \text{Re}[e(n)] = 0 \\ -1 & \text{Re}[e(n)] < 0 \end{cases}$$

The advantage of this algorithm is that the multiplication is replaced by a sign change, and if μ is chosen as a negative power of 2, then the multiplication is replaced by a shifting operation that is easy and fast to implement. Furthermore, since $\text{sgn}(x) = x/|x|$, the effective step size $\tilde{\mu}$ is inversely proportional to the magnitude of the error.

(a) Modify the function `firlms` to implement the sign-error NLMS algorithm and obtain the function

```
[c, e]=sefirlms(x, y, mu, M, c0)
```

(b) Repeat Problem 9.4 and compare your results in terms of convergence speed.

(c) Repeat Problem 9.5(*a*) and (*b*) and compare your results in terms of convergence speed.

9.8 Consider an AR(1) process $x(n) = ax(n-1) + \omega(n)$, where $\omega(n) \sim \text{WGN}(0,\ \sigma_\omega^2)$. We wish to design a one-step first-order linear predictor using the LMS algorithm

$$\hat{x}(n) = \hat{a}(n-1)\,x(n-1)$$
$$e(n) = x(n) - \hat{x}(n)$$
$$\hat{a}(n) = \hat{a}(n-1) + 2\mu e(n)\,x(n-1)$$

where μ is the adaptation step size.

(a) Determine the autocorrelation $r_x(l)$, the optimum first-order linear predictor, and the corresponding MMSE.

(b) Using the independence assumption, first determine and then solve the difference equation for $E\{\hat{a}(n)\}$.

(c) For $a = \pm 0.95$, $\mu = 0.025$, $\sigma_x^2 = 1$, and $0 \le n < N = 500$, determine the ensemble average of $E\{\hat{a}(n)\}$ using 200 independent runs and compare with the theoretical curve obtained in part (*b*).

(d) Using the independence assumption, first determine and then solve the difference equation for $P(n) = E\{e^2(n)\}$.

(e) Repeat part (*c*) for $P(n)$ and comment upon the results.

9.9 Using the a posteriori error $\varepsilon(n) = y(n) - \mathbf{c}^{\mathrm{H}}(n)\mathbf{x}(n)$, derive the coefficient updating formulas for the a posteriori error LMS algorithm. *Note*: Refer to Equations (9.2.20) to (9.2.22).

9.10 Solve the interference cancelation problem described in Example 5.4.1, using the LMS algorithm, and compare its performance to that of the optimum canceler.

9.11 Repeat the convergence analysis of the LMS algorithm for the complex case, using formula (9.4.27) instead of (9.4.28).

9.12 Consider the total transient excess MSE, defined by

$$P_{\mathrm{tr}}^{(\mathrm{total})} = \sum_{n=0}^{\infty} P_{\mathrm{tr}}(n)$$

in Section 9.4.3.

(a) Show that $P_{\mathrm{tr}}^{(\mathrm{total})}$ can be written as $P_{\mathrm{tr}}^{(\mathrm{total})} = \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{B})^{-1}\Delta\boldsymbol{\theta}(0)$, where $\Delta\boldsymbol{\theta}(0)$ is the initial (i.e., at $n = 0$) deviation of the filter coefficients from their optimum setting.

(b) Starting with the formula in step (*a*), show that

$$P_{\mathrm{tr}}^{(\mathrm{total})} = \frac{1}{4\mu} \frac{\sum_{i=1}^{M} \frac{\Delta\theta_i(0)}{1 - 2\mu\lambda_i}}{1 - \sum_{i=1}^{M} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i}}$$

(c) Show that if $\mu\lambda_k \ll 1$, then

$$P_{\mathrm{tr}}^{(\mathrm{total})} \simeq \frac{1}{4\mu} \frac{\sum_{i=1}^{M} \Delta\theta_i(0)}{1 - \mu\, \mathrm{tr}(\boldsymbol{R})} \simeq \frac{1}{4\mu} \sum_{i=1}^{M} \Delta\theta_i(0)$$

which is formula (9.4.62), discussed in Section 9.4.3.

9.13 The frequency sampling structure for the implementation of an FIR filter $H(z) = \sum_{n=0}^{M-1} h(n) \cdot z^{-n}$ is specified by the following relation

$$H(z) = \frac{1 - z^{-M}}{M} \sum_{k=0}^{M-1} \frac{H(\mathrm{e}^{\mathrm{j}2\pi k/M})}{1 - \mathrm{e}^{\mathrm{j}2\pi k/M} z^{-1}} \triangleq H_1(z) H_2(z)$$

where $H_1(z)$ is a comb filter with *M* zeros equally spaced on the unit circle and $H_2(z)$ is a filter bank of resonators. Note that $\tilde{H}(k) \triangleq H(\mathrm{e}^{\mathrm{j}2\pi k/M})$, the DFT of $\{h(n)\}_0^{M-1}$, provides the coefficients of the filter. Derive an LMS-type algorithm to update these coefficients, and sketch the resulting adaptive filter structure.

9.14 There are applications in which the use of a non-MSE criterion may be more appropriate. To this end, suppose that we wish to design and study the behavior of an "LMS-like" algorithm that minimizes the cost function $P^{(k)} = E\{e^{2k}(n)\}, k = 1, 2, 3, \cdots$, using the model defined in Figure 9.19.

(a) Use the instantaneous gradient vector to derive the coefficient updating formula for this LMS-like algorithm.

(b) Using the assumptions introduced in Section 9.4.2 show that

$$E\{\tilde{\boldsymbol{c}}(n)\} = [\boldsymbol{I} - 2\mu k(2k-1) E\{e_o^{2(k-1)}(n)\}\boldsymbol{R}] E\{\tilde{\boldsymbol{c}}(n-1)\}$$

where $\boldsymbol{R}$ is the input correlation matrix.

(c) Show that the derived algorithm converges in the mean if

$$0 < 2\mu < \frac{1}{k(2k-1)E\{e_o^{2(k-1)}(n)\}\lambda_{\max}}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{R}$.

(d) Show that for $k = 1$ the results in parts (*a*) to (*c*) reduce to those for the standard LMS algorithm.

9.15 Consider the noise cancelation system shown in Figure 9.6. The useful signal is a sinusoid $s(n) = \cos(\omega_0 n + \varphi)$, where $\omega_0 = \pi/16$ and the phase φ is a random variable uniformly distributed from 0 to 2π. The noise signals are given by $v_1(n) = 0.9v_1(n-1) + \omega(n)$ and $v_2(n) = -0.75v_2(n-1) + \omega(n)$, where the sequences $\omega(n)$ are WGN(0, 1).

(a) Design an optimum filter of length *M* and choose a reasonable value for M_o by plotting the MMSE as a function of *M*.

(b) Design an LMS filter with M_o coefficients and choose the step size μ to achieve a 10 percent misadjustment.

(c) Plot the signals $s(n)$, $s(n) + v_1(n)$, $v_2(n)$, the clean signal $e_o(n)$ using the optimum filter, and the clean signal $e_{\text{lms}}(n)$ using the LMS filter, and comment upon the obtained results.

9.16 A modification of the LMS algorithm, known as the *momentum LMS (MLMS)*, is defined by

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + 2\mu e^*(n)\boldsymbol{x}(n) + \alpha[\boldsymbol{c}(n-1) - \boldsymbol{c}(n-2)]$$

where $|\alpha| < 1$ (Roy and Shynk 1990).

(a) Rewrite the previous equation to show that the algorithm has the structure of a low-pass $(0 < \alpha < 1)$ or a high-pass $(-1 < \alpha < 0)$ filter.

(b) Explain intuitively the effect of the momentum term $\alpha[\boldsymbol{c}(n-1) - \boldsymbol{c}(n-2)]$ on the filter's convergence behavior.

(c) Repeat the computer equalization experiment in Section 9.4.4, using both the LMS and the MLMS algorithms for the following cases, and compare their performance:

i. $W = 3.1,\ \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.01,\ \alpha = 0.5$.

ii. $W = 3.1,\ \mu_{\text{lms}} = 0.04,\ \mu_{\text{mlms}} = 0.01,\ \alpha = 0.5$.

iii. $W = 3.1,\ \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.04,\ \alpha = 0.2$.

iv. $W = 4,\ \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.03,\ \alpha = 0.3$.

9.17 In Section 9.4.5 we presented the leaky LMS algorithm [see (9.4.88)]

$$\boldsymbol{c}(n) = (1 - \alpha\mu)\boldsymbol{c}(n-1) + \mu e^*(n)\boldsymbol{x}(n)$$

where $0 < \alpha \ll 1$ is the leakage coefficient.

(a) Show that the coefficient updating equation can be obtained by minimizing

$$P(n) = |e(n)|^2 + \alpha\|\boldsymbol{c}(n)\|^2$$

(b) Using the independence assumptions, show that

$$E\{\boldsymbol{c}(n)\} = [\boldsymbol{I} - \mu(\boldsymbol{R} + \alpha\boldsymbol{I})]E\{\boldsymbol{c}(n-1)\} + \mu\boldsymbol{d}$$

where $\boldsymbol{R} = E\{\boldsymbol{x}(n)\boldsymbol{x}^H(n)\}$ and $\boldsymbol{d} = E\{\boldsymbol{x}(n)y^*(n)\}$.

(c) Show that if $0 < \mu < 2/(\alpha + \lambda_{\max})$, where $\lambda_{\max}$ is the maximum eigenvalue of **R**, then

$$\lim_{n\to\infty} E\{\boldsymbol{c}(n)\} = (\boldsymbol{R} + \alpha\boldsymbol{I})^{-1}\boldsymbol{d}$$

that is, in the steady state $E\{\boldsymbol{c}(\infty)\} \neq \boldsymbol{c}_o = \boldsymbol{R}^{-1}\boldsymbol{d}$.

9.18 There are various communications and speech signal processing applications that require the use of filters with linear phase (Manolakis et al. 1984). For simplicity, assume that *m* is even.

(a) Derive the normal equations for an optimum FIR filter that satisfies the constraints

i. $\boldsymbol{c}_m^{(\text{lp})} = \boldsymbol{J}\boldsymbol{c}_m^{(\text{lp})}$ (linear phase)

ii. $\boldsymbol{c}_m^{(\text{cgd})} = -\boldsymbol{J}\boldsymbol{c}_m^{(\text{cgd})}$ (constant group delay).

(b) Show that the obtained optimum filters can be expressed as $\boldsymbol{c}_m^{(\text{lp})} = \frac{1}{2}(\boldsymbol{c}_m + \boldsymbol{J}\boldsymbol{c}_m)$ and $\boldsymbol{c}_m^{(\text{cgd})} = \frac{1}{2}(\boldsymbol{c}_m - \boldsymbol{J}\boldsymbol{c}_m)$, where $\boldsymbol{c}_m$ is the unconstrained optimum filter.

(c) Using the results in part (*b*) and the algorithm of Levinson, derive lattice-ladder structure for the constrained optimum filters.

(d) Repeat parts (*a*), (*b*), and (*c*) for the linear predictor with linear phase, which is specified by $\mathbf{a}_m^{(\text{lp})} = \mathbf{J}\mathbf{a}_m^{(\text{lp})}$.

(e) Develop an LMS algorithm for the linear-phase filter $\mathbf{c}_m^{(\text{lp})} = \mathbf{J}\mathbf{c}_m^{(\text{lp})}$ and sketch the resulting structure. Can you draw any conclusions regarding the step size and the misadjustment of this filter compared to those of the unconstrained LMS algorithm?

9.19 In this problem, we develop and analyze by simulation an LMS-type adaptive lattice predictor introduced in Griffiths (1977). We consider the all-zero lattice filter defined in (6.5.7), which is completely specified by the lattice parameters $\{k_m\}_0^{M-1}$. The input signal is assumed wide-sense stationary.

(a) Consider the cost function

$$P_m^{\text{fb}} = E\{|e_m^{\text{f}}(n)|^2 + |e_m^{\text{b}}(n)|^2\}$$

which provides the total prediction error power at the output of the *m*th stage, and show that

$$\frac{\partial P_m^{\text{fb}}}{\partial k_{m-1}^*} = 2E\{e_m^{\text{f}*}(n)e_{m-1}^{\text{b}}(n-1) + e_{m-1}^{\text{f}*}(n)e_m^{\text{b}}(n)\}$$

(b) Derive the updating formula

$$k_m(n) = k_m(n-1) - 2\mu(n)[e_m^{\text{f}*}(n)e_{m-1}^{\text{b}}(n-1) + e_{m-1}^{\text{f}*}(n)e_m^{\text{b}}(n)]$$

where the normalized step size $\mu(n) = \bar{\mu}/E_{m-1}^{\text{b}}(n)$ is computed in practice by using the formula

$$E_{m-1}^{\text{b}}(n) = \alpha E_{m-1}^{\text{b}}(n-1) + (1-\alpha)[|e_{m-1}^{\text{f}}(n)|^2 + |e_{m-1}^{\text{b}}(n-1)|^2]$$

where $0 < \alpha < 1$. Explain the role and proper choice of α, and determine the proper initialization of the algorithm.

(c) Write a MATLAB function to implement the derived algorithm, and compare its performance with that of the LMS algorithm in the linear prediction problem discussed in Example 9.4.1.

9.20 Consider a signal $x(n)$ consisting of a harmonic process plus white noise, that is,

$$x(n) = A\cos(\omega_1 n + \phi) + \omega(n)$$

where φ is uniformly distributed from 0 to 2π and $\omega(n) \sim \text{WGN}(0, \sigma_\omega^2)$.

(a) Determine the output power $\sigma_y^2 = E\{y^2(n)\}$ of the causal and stable filter

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

and show that we can cancel the harmonic process using the ideal notch filter

$$H(e^{j\omega}) = \begin{cases} 1 & \omega = \omega_1 \\ 0 & \text{otherwise} \end{cases}$$

Is the obtained ideal notch filter practically realizable? That is, is the system function rational? Why?

(b) Consider the second-order notch filter

$$H(z) = \frac{D(z)}{A(z)} = \frac{1 + az^{-1} + z^{-2}}{1 + a\rho z^{-1} + \rho^2 z^{-2}} = \frac{D(z)}{D(z/\rho)}$$

where $-1 < \rho < 1$ determines the steepness of the notch and $a = -2\cos\omega_0$ its frequency. We fix ρ, and we wish to design an adaptive filter by adjusting *a*.

i. Show that for $\rho \simeq 1$, $\sigma_y^2 = A^2 |H(e^{j\omega_1})|^2 + \sigma_\omega^2$, and plot σ_y^2 as a function of the frequency ω_0 for $\omega_1 = \pi/6$.

ii. Evaluate $d\sigma_y^2(a)/da$ and show that the minimum of $\sigma_y^2(a)$ occurs for $a = -2\cos\omega_1$.

(c) Using a direct-form II structure for the implementation of $H(z)$ and the property $dY(z)/da = [dH(z)/da]X(z)$, show that the following relations

$$s_2(n) = -a(n-1)\rho s_2(n-1) - \rho^2 s_2(n-2) + (1-gr)s_1(n-1)$$
$$g(n) = s_2(n) - \rho s_2(n-2)$$
$$s_1(n) = -a(n-1)\rho s_1(n-1) - \rho^2 s_1(n-2) + x(n)$$
$$y(n) = s_1(n) + a(n-1)s_1(n-1) + s_1(n-2)$$
$$a(n) = a(n-1) - 2\mu y(n)g(n)$$

constitute an adaptive LMS notch filter. Draw its block diagram realization.

(d) Simulate the operation of the obtained adaptive filter for $\rho = 0.9$, $\omega_1 = \pi/6$, and SNR 5 and 15 dB. Plot

$\omega_0(n) = \arccos[-a(n)/2]$ as a function of n, and investigate the tradeoff between convergence rate and misadjustment by experimenting with various values of μ.

9.21 Consider the AR(2) process given in Problem 9.4. We will design the adaptive linear predictor using the RLS algorithm. The adaptive predictor is a second-order one given by $\mathbf{c}(n) = [c_1(n)\ c_2(n)]^T$.
(a) Develop a MATLAB function to implement the RLS algorithm given in Table 9.6

```
[c, e]=rls(x, y, lambda, delta, M, c0)
```

which computes filter coefficients in `c` and the corresponding error in `e` given signal `x`, desired signal `y`, forgetting factor `lambda`, initialization parameter `delta`, filter order `M`, and the initial coefficient vector `c0`. To update $\mathbf{P}(n)$, compute only the upper or lower triangular part and determine the other part by using Hermitian symmetry.
(b) Generate 500 samples of $x(n)$ and obtain linear predictor coefficients using the above function. Use a very small value for δ (for example, 0.001) and various values of $\lambda = 0.99$, 0.95, 0.9, and 0.8. Plot predictor coefficients as a function of time along with the true coefficients for each λ, and discuss your observations. Also compare your results with those in Problem 9.4.
(c) Repeat each simulation above 1000 times to get corresponding learning curves, which are obtained by averaging respective squared errors $|e(n)|^2$. Plot these curves and compare their steady-state value with the theoretical MSE.

9.22 Consider a system identification problem where we observe the input $x(n)$ and the noisy output $y(n) = y_o(n) + v(n)$, for $0 \le n \le N-1$. The unknown system is specified by the system function

$$H_o(z) = \frac{0.0675 + 0.1349z^{-1} + 0.0675z^{-2}}{1 - 1.1430z^{-1} + 0.4128z^{-2}}$$

and $x(n) \sim \text{WGN}(0,1)$, $v(n) \sim \text{WGN}(0, 0.01)$, and $N = 300$.
(a) Model the unknown system using an LS FIR filter, with $M = 15$ coefficients, using the no-windowing method. Compute the total LSE E_{ls} in the interval $n_0 \le n \le N-1$ for $n_0 = 20$.
(b) Repeat part (*a*) for $0 \le n \le n_0 - 1$ (do not compute E_{ls}). Use the vector $\mathbf{c}(n_0)$ and the matrix $\mathbf{P}(n_0) = \hat{\mathbf{R}}^{-1}(n_0)$ to initialize the CRLS algorithm. Compute the total errors $E_{\text{apr}} = \sum_{n=n_0}^{N-1} e^2(n)$ and $E_{\text{apost}} = \sum_{n=n_0}^{N-1} \varepsilon^2(n)$ by running the CRLS for $n_0 \le n \le N-1$.
(c) Order the quantities $E_{\text{ls}}, E_{\text{apr}}, E_{\text{apost}}$ by size and justify the resulting ordering.

9.23 Prove Equation (9.5.25) using the identity $\det(\mathbf{I}_1 + \mathbf{AB}) = \det(\mathbf{I}_2 + \mathbf{BA})$, where identity matrices $\mathbf{I}_1$ and $\mathbf{I}_2$ and matrices $\mathbf{A}$ and $\mathbf{B}$ have compatible dimensions. *Hint:* Put (9.5.7) in the form $\mathbf{I}_1 + \mathbf{AB}$.

9.24 Derive the normal equations that correspond to the minimization of the cost function (9.5.36), and show that for $\delta = 0$ they are reduced to the standard set (9.5.2) of normal equations. For the situation described in Problem 9.22, run the CRLS algorithm for various values of δ and determine the range of values that provides acceptable performance.

9.25 Modify the CRLS algorithm in Table 9.6 so that its coefficients satisfy the linear-phase constraint $\mathbf{c} = \mathbf{J}\mathbf{c}^*$. For simplicity, assume that $M = 2L$; that is, the filter has an even number of coefficients.

9.26 Following the approach used in Section 6.5.1 to develop the structure shown in Figure 6.1, derive a similar structure based on the Cholesky (*not* the LDL^H) decomposition.

9.27 Show that the partitioning (9.6.3) of $\hat{\mathbf{R}}_{m+1}(n)$ to obtain the same partitioning structure as (9.6.2) is possible only if we apply the prewindowing condition $\mathbf{x}_m(-1) = \mathbf{0}$. What is the form of the partitioning if we abandon the prewindowing assumption?

9.28 Derive the normal equations and the LSE formulas given in Table 9.8 for the FLP and the BLP methods.

9.29 Derive the FLP and BLP a priori and a posteriori updating formulas given in Table 9.9.

9.30 Modify Table 9.11 for the FAEST algorithm, to obtain a table for the FTF algorithm, and write a MATLAB function for its

implementation. Test the obtained function, using the equalization experiment in Example 9.5.2.

9.31 If we wish to initialize the fast RLS algorithms (fast Kalman, FAEST, and FTF) using an exact method, we need to collect a set of data $\{\mathbf{x}(n), y(n)\}_0^{n_0}$ for any $n_0 > M$.

(a) Identify the quantities needed to start the FAEST algorithm at $n = n_0$. Form the normal equations and use the LDL^H decomposition method to determine these quantities.

(b) Write a MATLAB function `faestexact.m` that implements the FAEST algorithm using the exact initialization procedure described in part (*a*).

(c) Use the functions `faest.m` and `faestexact.m` to compare the two different initialization approaches for the FAEST algorithm in the context of the equalization experiment in Example 9.5.2. Use $n_0 = 1.5M$ and $n_0 = 3M$. Which value of δ gives results closest to the exact initialization method?

9.32 Using the order-recursive approach introduced in Section 6.3.1, develop an order-recursive algorithm for the solution of the normal equations (9.5.2), and check its validity by using it to initialize the FAEST algorithm, as in Problem 9.31. *Note:* In Section 6.3.1 we could not develop a closed-form algorithm because some recursions required the quantities $\boldsymbol{b}_m(n-1)$ and $E_m^b(n-1)$. Here we can avoid this problem by using time recursions.

9.33 In this problem we discuss several quantities that can serve to warn of ill behavior in fast RLS algorithms for FIR filters.

(a) Show that the variable

$$\eta_m(n) \triangleq \frac{\alpha_{m+1}(n)}{\alpha_m(n)} = \frac{\lambda E_m^{\text{b}}(n-1)}{E_m^{\text{b}}(n)} = 1 - g_{m+1}^{(m+1)}(n) e_m^{\text{b}*}(n)$$

satisfies the condition $0 \le \eta_m(n) \le 1$.

(b) Prove the relations

$$\alpha_m(n) = \lambda^m \frac{\det \hat{\boldsymbol{R}}_m(n-1)}{\det \hat{\boldsymbol{R}}_m(n)} \qquad E_m^{\text{f}}(n) = \frac{\det \hat{\boldsymbol{R}}_{m+1}(n)}{\det \hat{\boldsymbol{R}}_m(n-1)} \qquad E_m^{\text{b}}(n) = \frac{\det \hat{\boldsymbol{R}}_{m+1}(n)}{\det \hat{\boldsymbol{R}}_m(n)}$$

(c) Show that

$$\alpha_m(n) = \lambda^m \frac{E_m^{\text{b}}(n)}{E_m^{\text{f}}(n)}$$

and use it to explain why the quantity $\eta_m^{\alpha}(n) = E_m^{\text{f}}(n) - \lambda^m E_m^{\text{b}}(n)$ can be used as a warning variable.

(d) Explain how the quantities

$$\eta_{\bar{g}}(n) \triangleq \bar{g}_{M+1}^{(M+1)}(n) - \frac{e^{\text{b}}(n)}{\lambda E^{\text{b}}(n-1)}$$

and

$$\eta_b(n) \triangleq e^{\text{b}}(n) - \lambda E^{\text{b}}(n-1)\, \bar{g}_{M+1}^{(M+1)}(n)$$

can be used as warning variables.

9.34 When the desired response is $y(j) = \delta(j-k)$, that is, a spike at $j = k$, $0 \le k \le n$, the LS filter $\mathbf{c}_m^{(k)}$ is known as a *spiking filter* or as an LS inverse filter (see Section 7.3).

(a) Determine the normal equations and the LSE $E_m^{(k)}(n)$ for the LS filter $\boldsymbol{c}_m^{(k)}$.

(b) Show that $\boldsymbol{c}_m^{(n)} = \boldsymbol{g}_m(n)$ and $E_m^{(n)}(n) = \alpha_m(n)$ and explain their meanings.

(c) Use the interpretation $\alpha_m(n) = E_m^{(n)}(n)$ to show that $0 \le \alpha_m(n) \le 1$.

(d) Show that $\boldsymbol{a}_m(n) = \sum_{k=0}^{n} \boldsymbol{c}_m^{(k)}(n-1) x(k)$ and explain its meaning.

9.35 Derive Equations (9.6.33) through (9.6.35) for the a posteriori LS lattice-ladder structure, shown in Figure 9.37, starting with the partitionings (9.6.1) and the matrix by inversion by partitioning relations (9.6.7) and (9.6.8).

9.36 Prove relations (9.6.45) and (9.6.46) for the updating of the ladder partial correlation coefficient $\beta_m^c(n)$.

9.37 In Section 6.3.1 we derived order-recursive relations for the FLP, BLP, and FIR filtering MMSEs.

(a) Following the derivation of (6.3.36) and (6.3.37), derive similar order-recursive relations for $E_m^f(n)$ and $E_m^b(n)$.

(b) Show that we can obtain a complete LS lattice-ladder algorithm by replacing, in Table 9.12, the time-recursive updatings of $E_m^f(n)$ and $E_m^b(n)$ with the obtained order-recursive relations.

(c) Write a MATLAB function for this algorithm, and verify it by using the equalization experiment in Example 9.5.2.

9.38 Derive the equations for the a priori RLS lattice-ladder algorithm given in Table 9.13, and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 9.5.2.

9.39 Derive the equations for the a priori RLS lattice-ladder algorithm with error feedback (see Table 9.7), and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 9.5.2.

9.40 Derive the equations for the a posteriori RLS lattice-ladder algorithm with error feedback (Ling et al. 1986) and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 9.5.2.

9.41 The a posteriori and the a priori RLS lattice-ladder algorithms need the conversion factor $\alpha_m(n)$ because the updating of the quantities $E_m^f(n), E_m^b(n), \beta_m(n)$, and $\beta_m^c(n)$ requires both the a priori and a posteriori errors. Derive a double (a priori and a posteriori) lattice-ladder RLS filter that avoids the use of the conversion factor by updating both the a priori and the a posteriori prediction and filtering errors.

9.42 The implementation of adaptive filters using multiprocessing involves the following steps: (1) partitioning of the overall computational job into individual tasks, (2) allocation of computational and communications tasks to the processors, and (3) synchronization and control of the processors. Figure 9.52 shows a cascade multiprocessing architecture used for adaptive filtering. To avoid *latency* (i.e., a delay between the filter's input and output that is larger than the sampling interval), each processor should complete its task in time less than the sampling period and use results computed by the preceding processor and the scalar computational unit at the previous sampling interval. This is accomplished by the unit delays inserted between the processors.

(*a*) Explain why the fast Kalman algorithm, given in Table 9.10, does not satisfy the multiprocessing requirements.

(*b*) Prove the formulas

$$\boldsymbol{b}(n) = \frac{\boldsymbol{b}(n-1) - \boldsymbol{g}_{M+1}^{[M]}(n)e^{b*}(n)}{1 - g_{M+1}^{(M+1)}(n)e^{b*}(n)} \tag{k}$$

$$\boldsymbol{g}(n) = \boldsymbol{g}_{M+1}^{[M]}(n) - g_{M+1}^{(M+1)}(n)\boldsymbol{b}(n) \tag{l}$$

and show that they can be used to replace formulas (g) and (h) in Table 10.

(c) Rearrange the formulas in Table 9.10 as follows: (*e*), (*k*), (*l*), (*a*), (*b*), (*c*), (*f*), (*d*), Replace *n* by $n-1$ in (*e*), (*l*), and (*k*). Show that the resulting algorithm complies with the multiprocessing architecture shown in Figure 9.52.

(*d*) Draw a block diagram of a single multiprocessing section that can be used in the multiprocessing architecture shown in Figure 10.52. Each processor in Figure 9.52 can be assigned to execute one or more of the designed sections. Note: You may find useful the discussions in Lawrence and Tewksbury (1983) and in Manolakis and Patel (1992).

(*e*) Figure 9.53 shows an alternative implementation of a multiprocessing section that can be used in the architecture of Figure 9.52. Identify the input-output quantities and the various multiplier factors.

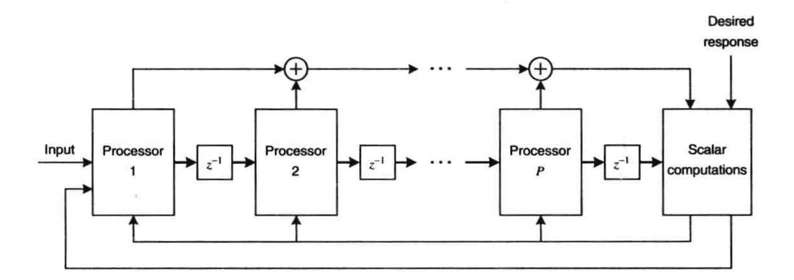


FIGURE 9.52
Cascade multiprocessing architecture for the implementation of FIR adaptive filters.

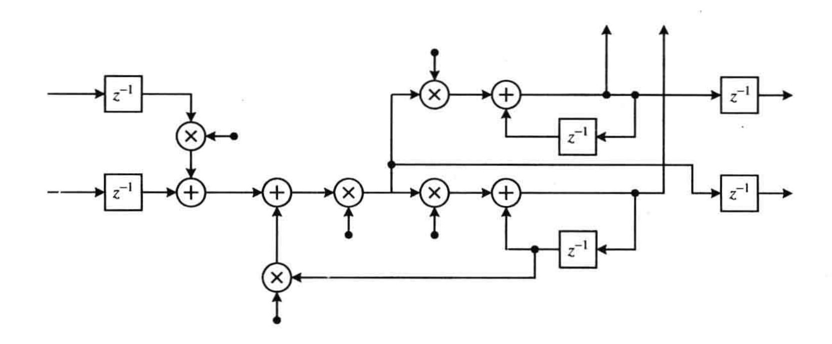


FIGURE 9.53
Section for the multiprocessing implementation of the fast Kalman algorithm.

9.43 Repeat Problem 9.48 for the FAEST algorithm shown in Table 9.11.

9.44 Show that the a priori RLS linear prediction lattice (i.e., without the ladder part) algorithm with error feedback complies with the multiprocessing architecture of Figure 9.52. Explain why the addition of the ladder part violates the multiprocessing architecture. Can we rectify these violations? (See Lawrence and Tewksbury 1983.)

9.45 The fixed-length sliding window RLS algorithm is given in (9.7.4) through (9.7.10).
(a) Derive the above equations of this algorithm (see Manolakis et al. 1987).
(b) Develop a MATLAB function to implement the algorithm

```
[c,e]=slwrls(x, y, L, delta, M, c0);
```

where L is the fixed length of the window.
(c) Generate 500 samples of the following nonstationary process

$$x(n) = \begin{cases} \omega(n) + 0.95x(n-1) - 0.9x(n-2) & 0 \le n < 200 \\ \omega(n) - 0.95x(n-1) - 0.9x(n-2) & 200 \le n < 300 \\ \omega(n) + 0.95x(n-1) - 0.9x(n-2) & n \ge 300 \end{cases}$$

where $\omega(n)$ is a zero-mean, unit-variance white noise process. We want to obtain a second-order linear predictor using adaptive algorithms. Use the sliding window RLS algorithm on the data and choose $L = 50$ and 100. Obtain plots of the filter coefficients and mean square error.
(d) Now use the growing memory RLS algorithm by choosing $\lambda = 1$. Compare your results with the sliding-window RLS algorithm.
(e) Finally, use the exponentially growing memory RLS by choosing $\lambda = (L-1)/(L+1)$ that produces the same MSE. Compare your results.

9.46 Consider the definition of the MSD $\mathcal{D}(n)$ in (9.2.29) and that of the trace of a matrix.
(a) Show that $D(n) = tr\{\boldsymbol{\Phi}(n)\}$, where $\boldsymbol{\Phi}(n)$ is the correlation matrix of $\tilde{\boldsymbol{c}}(n)$.
(b) For the evolution of the correlation matrix in (9.7.58), show that

$$\mathcal{D}(\infty) \simeq \mu M \sigma_v^2 + \frac{\text{tr}(\boldsymbol{R}^{-1}\boldsymbol{R}_\psi)}{4\mu}$$

9.47 Consider the analysis model given in Figure 9.39. Let the parameters of this model be as follows:

$\mathbf{c}_o(n)$ model parameters: $\quad \mathbf{c}_o(0) = \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix} \quad M = 2 \quad \rho = 0.95$

$\quad \boldsymbol{\psi}(n) \sim \text{WGN}(\boldsymbol{0}, \boldsymbol{R}_\psi) \quad \boldsymbol{R}_\psi = (0.01)^2 \boldsymbol{I}$

Signal $\mathbf{x}(n)$ parameters: $\quad \boldsymbol{x}(n) \sim \text{WGN}(\boldsymbol{0}, \boldsymbol{R}) \quad \boldsymbol{R} = \boldsymbol{I}$

Noise $v(n)$ parameters: $\quad v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the LMS algorithm.

(a) Obtain the tracking plots similar to Figure 9.40 for each of the above three adaptations.
(b) Obtain the learning curve plots similar to Figure 9.41 for each of the above three adaptations.

9.48 Consider the analysis model given in Figure 9.39. Let the parameters of this model be as follows

$\mathbf{c}_o(n)$ model parameters: $\mathbf{c}_o(0) = \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix}$ $M = 2$ $\rho = 0.95$

$\boldsymbol{\psi}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi)$ $\mathbf{R}_\psi = (0.01)^2 \mathbf{I}$

Signal $\mathbf{x}(n)$ parameters: $\mathbf{x}(n) \sim \text{WGN}(0, \mathbf{R})$ $\mathbf{R} = \mathbf{I}$

Noise $v(n)$ parameters: $v(n) \sim \text{WGN}(0, \sigma_v^2)$ $\sigma_v = 0.1$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the RLS algorithm.

(a) Obtain the tracking plots similar to Figure 9.46 for each of the above three adaptations.
(b) Obtain the learning curve plots similar to Figure 9.47 for each of the above three adaptations.
(c) Compare your results with those obtained in Problem 9.53.

9.49 Consider the time-varying adaptive equalizer shown in Figure 9.54 in which the time variation of the channel impulse response is given by

$$h(n) = \rho h(n-1) + \sqrt{1-\rho}\,\eta(n)$$

with $\rho = 0.95$ $\eta(n) \sim \text{WGN}(0, \sqrt{10})$ $h(0) = 0.5$

Let the equalizer be a single-tap equalizer and $v(n) \sim \text{WGN}(0, 0.1)$.

(a) Simulate the system for three different adaptations; that is, choose μ for slow, matched, and fast adaptations of the LMS algorithm.
(b) Repeat part (*a*), using the RLS algorithm.

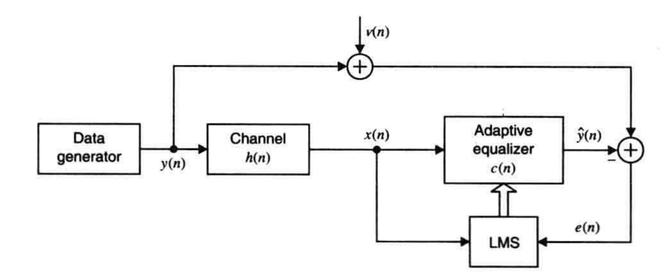


FIGURE 9.54
Adaptive channel equalizer system with time-varying channel in Problem 9.55.